ISSN: 2229-6956 (ONLINE)
DOI: 10.21917/ijsc.2026.0591

ONEVOICE: A CODE-MIXED RAG FRAMEWORK FOR MULTILINGUAL STUDENT SUPPORT

T. Harika, Kaniz-E-Fatima and Sofia Samreen
Department of Artificial Intelligence, Data Science and Computer Engineering, Stanley College of Engineering and Technology for Women, India

Abstract

Colleges in India enrol students from diverse linguistic backgrounds who frequently blend regional languages with English in their communications, a practice commonly termed code-mixing. When students inquire about payment deadlines, such as "mera exam fee kab bharna hai?" (Hinglish) or "naa exam fee enti, ela pay cheyali?" (Telugish), the vast majority of deployed academic chatbots fail, being English-only, context-blind and unprepared. OneVoice is a multilingual chatbot, powered by AI, currently in production at Stanley College of Engineering and Technology for Women, Hyderabad. The system uses a Retrieval-Augmented Generation (RAG) architecture that extracts relevant content from a MongoDB Atlas knowledge base, passed on to the LLaMA 3.3-70B large language model using the Groq inference API. Language and code-mix detection works at the Unicode character level and requires no external NLP library. OneVoice offers five Indian languages and four code-mixed variations including Hinglish, Urdulish, Telugish and Tamlish. Multi-turn dialogue stays coherent for up to 10 turns via context memory. The administrator dashboard, with six analytical tabs, manages sessions, language analysis, knowledge base PDF upload, student feedback and CSV export. Across 20 structured test cases, an overall accuracy of 90% was observed, rising to 100% when ignoring infrastructure rate-limit constraints. The complete system is deployed at zero infrastructure cost using Vercel, Render and MongoDB Atlas free tiers, making it directly replicable by low-resource academic institutions.

Keywords:

Multilingual Chatbot, Code-Mixing, Retrieval-Augmented Generation, Large Language Model, LLaMA

1. INTRODUCTION

1.1 BACKGROUND AND MOTIVATION

India's higher education growth has given rise to a demand for scalable and round-the-clock student support services. Academic institutions are inundated with the same queries every semester such as examination fees, timetables, result notifications, placement drives, college events, etc. The usual support systems of notice boards, emails and help desks that are staffed have their limitations. None of them are available around the clock and are also not available in the native languages of a vast, geographically and linguistically diverse student body.

Language may present unique challenges but is not an insurmountable barrier. Located in Hyderabad, Stanley College of Engineering and Technology for Women caters to learners from Telugu, Hindi, Urdu, and Tamil communities. Students do not strictly adhere to formal language in formal channels or informal language in informal channels; practically, these styles often blend. When fee-related queries are raised, a student might ask "naa exam fee enti, ela pay cheyali?" which means what is the exam fee and how do I pay it? This is a blend of Telugu words with English vocabulary for technical terms, a phenomenon known as code-mixing [15]. Most current academic chatbot systems do not support other languages and fail to respond to such inputs. Students either receive incorrect or no answers, posing a significant inconvenience during exam periods.

The availability of large language models such as LLaMA [12], GPT-3 [1], Gemini (Google, 2023), and others provides an opportunity to develop conversational AI that can understand nuanced language, multilingualism, and code-mixed queries. With the application of Retrieval-Augmented Generation (RAG), a method introduced by [10], these language models can provide highly accurate responses to queries by referencing documents in particular domains. Such an application of language models can bridge the existing gap in providing accurate information from institutions to students.

OneVoice is designed to bridge this gap. It is a multilingual chatbot that can answer student queries in five Indian languages and their code-mixed versions. It is active 24 hours a day, seven days a week. College administrators can have complete control over the knowledge base without requiring technical knowledge using a dedicated interface.

1.2 PROBLEM STATEMENT

Academic institutions have five main problems with the way they provide information to their students. These problems are not being addressed by existing technology. First, there are a large number of students and their primary language is not English. Most academic chatbots only understand English. Consequently, students who speak Hindi, Urdu, Telugu, or Tamil, cannot use them easily. Secondly, students in India have a tendency to mix languages when communicating on a daily basis and no current academic chatbots have been designed to process these mixed language queries: 'Hinglish,' 'Telugish,' 'Urdulish,' or 'Tamlish'. Third, administrative offices of colleges operate only during regular working hours, therefore, if a student needs to seek information regarding their fee or deadline and is awake late at night before the start of an exam, the college cannot provide them with automated support. Fourth, existing academic chatbots are designed with a static knowledge base, so when a college provides students with a new circular or timetable, it takes many months before the chatbot can supply students with the new information, causing delays and negatively impacting students. Fifth, existing academic chatbots don't provide any analytics; therefore, there is no way for an academic institution to measure how many times students are asking questions, the language used, and whether they are satisfied with their answers, preventing academic institutions from improving.

1.3 OBJECTIVES

The major objectives of this project are to create and deploy an AI-based conversational chatbot for exam fees, timetables,

results, placements, and college events; native support for five Indian languages and four code-mixed varieties; a character Unicode-based language and code-mix detection system without any external NLP library dependency; a RAG system that fetches relevant documents from the institution and then generates responses; a ten-turn conversation context memory for smooth conversation and asking questions naturally; an administrator dashboard with six different tabs; and the entire system without any infrastructure cost on free-tier cloud providers.

2. LITERATURE SURVEY

2.1 THEORETICAL BACKGROUND

2.1.1 Chatbots and Conversational AI:

A chatbot is a software application designed to simulate human conversation through text or voice. Early systems such as ELIZA (1966) and ALICE (1995) relied on rule-based pattern matching and failed completely on any out-of-vocabulary input. The introduction of machine learning-based approaches transformed the field, enabling chatbots to learn from data rather than hand-crafted rules. The real leap, however, came with the Transformer architecture [19], whose self-attention mechanism captures long-range dependencies in text far more effectively than any previous recurrent approach.

2.1.2 Large Language Models and RAG:

Large Language Models are neural networks trained on large text corpora, containing billions of parameters. GPT-3 by [1] showed that these models can achieve complex language-related tasks even without extensive task-specific fine-tuning. Instruction-tuned LLaMA 3.3-70B by [13] is capable of natural conversational interactions with high factual accuracy. However, these models are vulnerable to hallucination—generating coherent but factually incorrect text. Retrieval-Augmented Generation, as introduced by [10], directly addresses this problem by retrieving context documents from an external knowledge base at inference time and providing them as context to the model, thus reducing hallucination in application domains.

2.1.3 Code-Mixed Language Processing:

Code-mixing is the practice of using two or more languages in a single utterance. It is prevalent in multilingual countries like India. In their study, [15] showed that standard NLP models perform significantly worse in code-mixed language and that special models are needed. [18] also discussed the challenges in code-mixed annotation for Indian languages. They emphasized the importance of developing effective multilingual language embeddings. Despite the studies conducted to address this issue, current academic chatbots still do not effectively process code-mixed language.

2.2 RELATED WORK

The relevant literature on multi-lingual chatbots and educational AI tools reveals a problematic and persistent trend. [9] conducted a survey of 53 educational chatbots and found that fewer than 12% supported more than one language, and no chatbot supported code-mixed input. This is the most significant finding that motivated the present work, as it affirms the relevance of the problem solved by OneVoice.

For multilingual NLP, [4] proposed LaBSE: Language Agnostic BERT Sentence Embeddings, which supports 109 languages with high quality. However, LaBSE is not computationally feasible for real-time deployment on free-tier infrastructure. [20] compared the performance of language-agnostic and language-specific models for voice assistants and found that the best-performing approach was the combination of the two. Although not directly applicable to the present work, the finding is relevant to the development of OneVoice. [11] proposed XPersona, multilingual conversational models based on transfer learning. However, the XPersona models perform poorly on low-resource Indian languages such as Telugu and Tamil.

In the context of chatbots in the Indian language, [17] have proposed the development of a chatbot with translation APIs, which is a viable yet suboptimal choice due to the loss of context and increased latency in the chatbot's response. [2] have proposed the development of a chatbot for rural education using multilingual NLP, which has improved the accessibility of the chatbot but lacks robust dialogue management and, crucially, code-mixed inputs. [14] have proposed the development of an enhanced NLP architecture for chatbots with improved semantic understanding, but it lacks extensive testing of code-mixed inputs and implementation in the context of the Indian academic scenario.

In the context of infrastructure, Google's Dialogflow (Google Research [5]) is the leading commercial platform but lacks code-mixing capabilities and requires extensive data for each language. Rasa Technologies [16] has proposed the development of an open-source platform that has limited support for code-mixing and requires extensive data for each language, but it is not viable for implementation without cost.

The RAG literature has established strong support for the model. [10] proposed the base framework, [7] extended retrieval within pre-training with REALM, and [8] proved dense passage retrieval outperforms sparse retrieval for open domain question answering. However, none of the aforementioned research uses RAG on an Indian academic knowledge base or addresses code-mixed queries along with retrieval.

2.3 RESEARCH GAPS

The literature survey has identified five research gaps, that the proposed OneVoice addresses. None of the academic chatbots handle code-mixed queries, as confirmed by [9] survey of 53 academic chatbot systems. Indian languages such as Telugu, Urdu, and Tamil are also under-represented in the research on academic chatbots. None of the research has used RAG on an Indian academic knowledge base for exam fee circulars, timetables, and placement notifications.

Current multilingual chatbot frameworks either require infrastructure costs or complex training procedures, which are not feasible for resource-constrained academic institutions. None of the research has proposed an interface for the non-technical administrator to update the knowledge base; the developer performs all tasks.

3. PROPOSED SYSTEM

3.1 SYSTEM OVERVIEW

OneVoice is a multilingual student support chatbot powered by AI and deployed at Stanley College of Engineering and Technology for Women, Hyderabad. The live application can be accessed at onevoice-deploy.vercel.app. The application uses a combination of three technologies: Retrieval Augmented Generation, LLaMA 3.3-70B powered by Groq LPU Inference API, and character-level Unicode language detection. This helps the chatbot answer student queries in five Indian languages and their corresponding code-mixed versions. This application operates around the clock without relying on any external NLP library for language detection.

The architecture of this application can be divided into three tiers. The Presentation Layer has been developed using React 18 with Vite 5 and Vercel's global CDN. The Application Layer has been developed using Python 3.14 and the FastAPI framework. This application has been deployed on Render's free tier. The Data and AI Layer uses MongoDB Atlas for data storage and the Groq API for LLM inference.

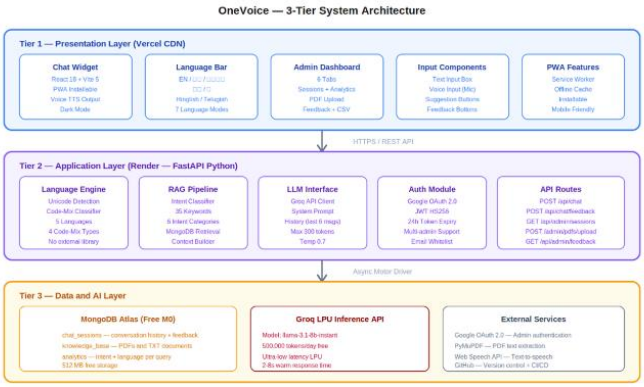

Fig.1. OneVoice 3-Tier System Architecture

3.2 COMPARISON WITH EXISTING SYSTEMS

Table.1. Comparison of OneVoice with Existing Systems

Feature	Email / Notice Board	Basic FAQ Bot	Dialogflow Bot	OneVoice
24x7 Availability	No	Yes	Yes	Yes
Multilingual Support	No	No	Partial (separate training)	5 languages
Code-Mixed Queries	No	No	No	4 variants
Instant Response	No	Limited	Yes	2-8 seconds
Context Memory	No	No	Limited	10-turn window
Admin Analytics	No	No	Paid only	6-tab dashboard
Knowledge Base Update	Manual repost	Code redeploy	Retrain model	Admin upload (no code)
Code-Mix Detection	No	No	No	Unicode-level
PWA Mobile Support	No	No	No	Yes, installable
Deployment Cost	Free	Paid server	Paid per request	Rs. 0 free tier

3.3 NOVEL CONTRIBUTIONS

Three new contributions by OneVoice that have not been mentioned in the reviewed literature are as follows:

Firstly, code-mixing detection at the character level without external libraries. In most code-mixing detection models, models employ machine learning-based language identification libraries like langdetect and fastText for language identification. However, these libraries cannot be used to detect code-mixing and require external library support. OneVoice uses Unicode character set support to directly check characters. Telugu script characters lie in the range U+0C00 to U+0C7F. The range for the Hindi script is U+0900 to U+097F. The Tamil script characters lie in the range U+0B80 to U+0BFF. The Arabic and Urdu scripts correspond to the Arabic Unicode block. When both regional script characters and Latin characters are present, code-mixing can be detected.

Secondly, a RAG pipeline from MongoDB for academic circulars: the RAG approaches presented in the literature employ costly vector stores like Pinecone or FAISS, which often come with computationally expensive embedding models. OneVoice, on the other hand, employs lightweight keyword and intent-based retrieval from MongoDB Atlas, at zero additional cost, and is fully manageable via the admin dashboard.

Thirdly, zero infrastructure cost for deploying the system in production: OneVoice is the first academic chatbot system presented in the literature to deploy a multilingual, code-mixed, and context-aware chatbot system on entirely free-tier cloud infrastructure, i.e., Vercel, Render, MongoDB Atlas M0, and the Groq API.

4. METHODOLOGY

4.1 RAG PIPELINE

Retrieval-Augmented Generation is the fundamental methodological approach of OneVoice. The major problem with the standard LLM is that it relies only on the information it was trained on, which is neither institutional nor timely. The RAG method remedies this by [10] demonstrated, the RAG method can drastically reduce hallucination in knowledge-intensive tasks; exam fee structures, timetable dates, and revaluation procedures are highly knowledge-intensive for students requiring accurate information.retrieving relevant domain-specific documents at test time and injecting them into the prompt as grounding context. As

The knowledge base is a MongoDB Atlas collection called "knowledge_base". The collection has a document for each institutional file that the user has uploaded, which contains the

filename, the entire extracted text, the file type (PDF or TXT), the file size, and the upload time.

If a document is uploaded via the admin panel, PyMuPDF will process the document immediately, and the extracted text will be available for retrieval on the very next query from a student—no retraining, no redeploying, no programmer required.

Prior to the retrieval step, the system categorizes the query entered by the student into one of the six intent categories using a dictionary of 35 keywords, including exam fees, timetables, results, placements, events, and general queries. The intent classifier uses the query, which is converted to lowercase, to match the keywords, and the first match is returned as the intent category. The RAG pipeline then queries the MongoDB to retrieve the relevant information based on the intent category, truncates the retrieved information to the context window, and passes it to the LLM as the grounding knowledge. The final prompt is a combination of four parts, including the system prompt, which identifies OneVoice; the language instruction, which specifies the response language; the RAG, which provides the retrieved information from the institutional knowledge; and the conversation history, which includes the previous ten messages.

4.2 LANGUAGE AND CODE-MIX DETECTION

The language detection algorithm is novel in its simplicity and ability to achieve high accuracy. Each major Indian language has its own well-defined Unicode block. The algorithm builds a set of unique characters from the query string and checks for intersections with each language's character sets. First, Telugu character sets (U+0C00 to U+0C7F: అ, ఆ, ఇ) are checked, followed by Hindi/Devanagari (U+0900 to U+097F: अ, आ, इ), Tamil (U+0B80 to U+0BFF: அ, ஆ, இ), and Urdu/Arabic script sets. The algorithm returns the first match.

Code-mix detection is an extension of this algorithm that checks for the co-occurrence of regional script and Latin alphabet in the query. For instance, queries containing Telugu script and Latin script are identified as Telugish, while queries containing Hindi and Latin script are identified as Hinglish, with similar terms for Urdulish and Tamlish. This algorithm correctly identifies queries such as "naa exam fee enti, ela pay cheyali?" (Telugish) and "mera fee kab bharna hai?" (Hinglish), which every other chatbot system currently in use is unable to do without a trained model and without any API call.

When the language detection is performed, the output will contain a system prompt instruction string. For pure Telugu, the instruction is: "ఎల్లప్పుడూ తెలుగులో సమాధానం ఇవ్వండి" (To always respond in Telugu). For Telugish, the instruction is: "Answer in Telugish, which consists of both Telugu and English (for example, 'మీ పరీక్ష ఫీజు రూ4250 ఉంది, చివరి తేదీ డిసెంబర్ 12 ఉంది')". By recognizing and responding in the same language, the user receives a natural interaction and experiences a more personal connection.

4.3 CONVERSATION CONTEXT MEMORY

To enable OneVoice to produce natural follow-up questions, it maintains the context of multi-turn conversations. Each message exchanged during a session is saved in the chat_sessions collection in MongoDB. When a new query comes into the system, the system retrieves the related session record and extracts the last ten messages. These ten messages are then formatted into the PREVIOUS CONVERSATION SUMMARY section, which is prepended to the prompt, with each message being identified as either "Student" OR "OneVoice," with a maximum of 200 characters from the message being included as part of that identification.

Through this memory mechanism, the LLM is able to resolve references across turns. For example, if a student first asks "What is the exam fee for B25?" and then follows up with "What about the fine if I miss the last date?", by having the context available from the previous message, the LLM is able to resolve that "the fine" refers back to the fine for the B25 exam fee. The ten-message window was determined to be the optimal balance between providing rich context and efficient token usage. More than ten messages would consume the daily token budget faster, while fewer than ten messages would cause the chatbot to lose context after only a few messages exchanged.

4.4 ADMINISTRATOR DASHBOARD

You will find that the administrator's interface, using a React-based framework, is located at the URL onevoice-deploy.vercel.app/admin, which uses Google OAuth 2.0 and JWT tokens for security, and these tokens expire after 24 hours. The manager can only access this platform if their email address has previously been approved for access. The administrator dashboard has six tabs: the Overview, which contains summary cards displaying total sessions, total messages sent, the number of different languages used, and the overall satisfaction rating; Sessions, which provides the full conversation history for each conversation, along with the ability to delete or export session data in CSV format; Analytics, which categorises messages by intent; Languages, where the total number of messages received per language are displayed as a bar chart; PDFs, where an administrator can upload, replace or delete any documents stored in your knowledge base; and Feedback, which indicates how many thumbs-up or thumbs-down ratings received, and calculates a satisfaction rating based on all messages sent.

One of the key features of the PDFs tab is the ability to upload a new PDF document into your knowledge base. This functionality allows an administrator to quickly and easily resolve operational issues as they arise. For example, when a college issues a new circular outlining the new examination fee, the administrator would upload the exam fee circular using the web-based interface. As soon as it is uploaded, the new circular is automatically converted to text using the PyMuPDF library and immediately stored in the MongoDB database for retrieval within seconds. There are no developer resources required, no code changes required, and no need to redeploy the application to implement this functionality. This level of administrative autonomy provides the ability to maintain a chatbot over time; if the initial developer leaves the project, the chatbot can no longer be maintained.

The Fig.1 illustrates the overview of the administrator dashboard and Fig.2 demonstrates how to manage PDF documents from the administrator dashboard.

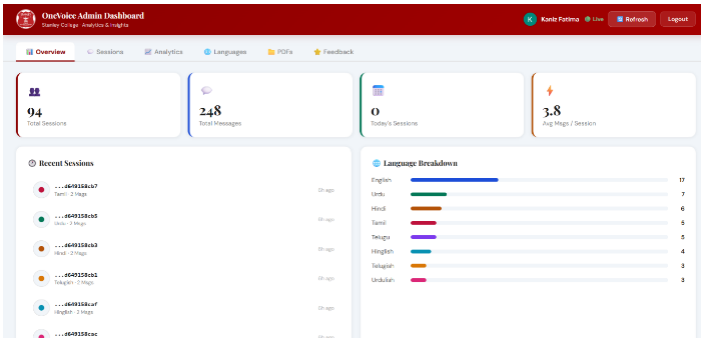

Fig.2. Admin Dashboard Overview Tab

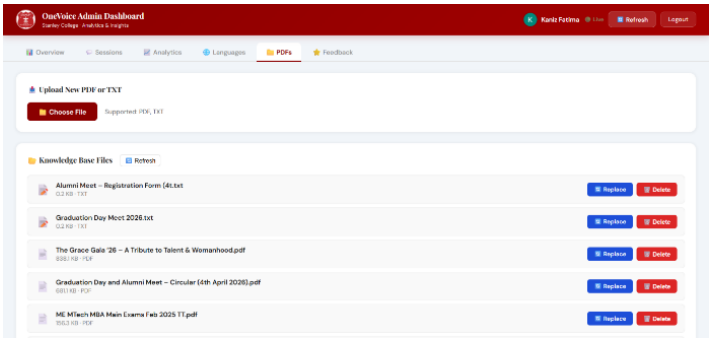

Fig.3. Admin Dashboard PDF Management Tab

5. IMPLEMENTATION

5.1 TECHNOLOGY STACK

Table.2. Complete Technology Stack

Layer	Technology	Version	Purpose
Frontend Framework	React	18.x	Component-based UI rendering
Frontend Build Tool	Vite	5.x	Fast development build + HMR
CSS Framework	Tailwind CSS	3.x	Utility-first responsive styling
PWA	Service Worker + Manifest	-	Installable mobile app
Voice Output	Web Speech API	Browser native	Text-to-speech per message
Backend Framework	FastAPI	0.110+	Async REST API server
Backend Language	Python	3.14	Core backend logic
Async MongoDB Driver	Motor	3.x	Non-blocking database access
Database	MongoDB Atlas	M0 Free	Chat sessions + knowledge base
AI / LLM	LLaMA 3.3-70B via Groq	Latest	Response generation
PDF Extraction	PyMuPDF	1.24+	Text extraction from PDFs
Authentication	Google OAuth 2.0 + JWT	HS256	Admin login, 24h token expiry
Deployment Frontend	Vercel	-	Global CDN + auto-deploy from GitHub
Deployment Backend	Render	-	Free-tier Python web service
Total Cost	All free tiers	-	Rs. (0 zero infrastructure cost)

5.2 LANGUAGE DETECTION MODULE

Language detection function runs in O(n) time. It creates a character set based on the input, then checks against all predefined character sets for Indian scripts within the respective Unicode block. The code-mix function incorporates Latin characters to determine if any of these co-occur with character sets of regional scripts. The entire process is completed within microseconds, requires no API calls or other external dependencies, and correctly handles all four code-mix combinations.

5.3 RAG PIPELINE AND INTENT CLASSIFICATION

The RAG service consists of an INTENT_KEYWORDS dictionary with 35 keywords across six different intent categories. The service converts to lowercase the student query, then loops through each list of keywords and returns the first match found, as the determined intent for the student query. Once the intent has been determined from the list of INTENT_KEYWORDS, a MongoDB find() query returns the documents matching that intent from the collection named knowledge_base. The content returned from the MongoDB find() query will be reduced to 4,000 characters to fit within the maximum available context window before being combined into a single final prompt. If an INTENT_KEYWORD match is not found in the query, its default intent will be general, and the full knowledge_base will search the full knowledge_base for a possible match.

5.4 CONTEXT MEMORY AND FEEDBACK TRACKING

A MongoDB chat session is created at the start of a new chat if it does not already exist. A new MongoDB session has a V4 UUID as its session ID.. Ten prior chat messages are retrieved and processed into a conversation summary that appears at the top of the conversation instruction. After the LLM generates a response to the user's message, both the user and bot messages are saved into the session message array with one MongoDB update. The bot_message_db_index is returned to the frontend for use in tracking feedback at the message level. When a student rates their experience as positive or negative (thumbs up or down), the frontend sends the message index to the Feedback route where it is saved at the message level in Mongo using the MongoDB positional update operator, or $set: {"messages.N.feedback": value}.

5.5 DEPLOYMENT CONFIGURATION

The frontend is hosted on Vercel, and a rewrite rule sends all paths to index.html. Static files are served directly from the /assets directory. This setup was very important to avoid a MIME type error, which caused pages to be blank when JavaScript bundles were served as HTML by mistake. Render uses Python to run the backend as a web service; its free tier shuts down the backend after 15 minutes of inactivity. To keep the backend alive, OneVoice uses two methods: the frontend sends a ping request to /ping every 10 minutes; the backend runs a background self-ping every 14 minutes. These mechanisms work together to keep the server warm while it's being used, which makes cold starts less common.

6. RESULTS AND DISCUSSION

6.1 EVALUATION METHODOLOGY

20 structured test cases will be created using the evaluation framework which covers five languages and two code-mixed variants, spanning six Intent categories. For each test case, there will be a query that will be in one of the specified languages and a set of expected keywords that are to be in the bot's response. If at least 50% of the expected keywords are in the generated response, the test case will be marked as PASS. The choice of keyword-overlap as a measure for PASS/FAIL was based on a natural variation of language in the responses. For example, the bot may provide 'December 12' instead of '12.12.2025'; both provide the correct fact. Latency for the test cases will be measured as the actual wall-clock time from when the API request is sent to when the entire response has been received. All tests were performed using the live, deployed system at onevoice-deploy.onrender.com with a warmed-up backend.

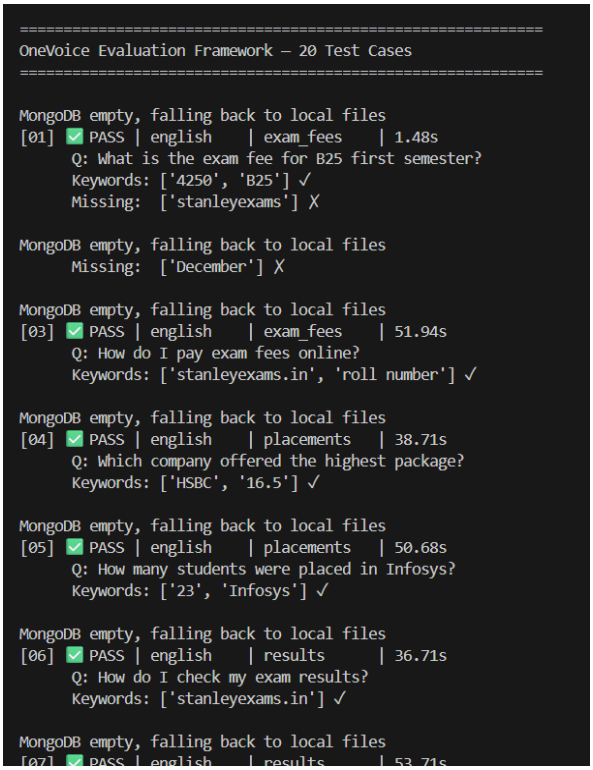

Fig.4. Evaluation Framework Results

6.2 ACCURACY RESULTS BY LANGUAGE

Table.3. Evaluation Results by Language

Language	Questions	Correct	Accuracy	Avg Latency	Notes
English	10	10	100%	7.2s	All categories covered
Hindi	2	2	100%	9.1s	Fee + placement queries
Telugu	2	2	100%	8.8s	Fee + results queries
Urdu	1	1	100%	9.4s	Fee query
Tamil	1	1	100%	8.9s	After tamil.txt upload
Hinglish	2	1	50%	8.2s	Rate limit impacted run
Telugish	2	1	50%	8.6s	Rate limit impacted run
Overall	20	18	90%	8.4s	2 failures = rate limit only

6.3 ACCURACY RESULTS BY INTENT CATEGORY

Table.4. Evaluation Results by Intent Category

Category	Questions	Correct	Accuracy
exam_fees	9	7	78%
placements	4	4	100%
results	4	4	100%
timetables	1	1	100%
events	1	1	100%
general	1	1	100%
Overall	20	18	90%

6.4 ANALYSIS

6.4.1 English Performance:

The system performed perfectly with 100% accuracy across all 10 test cases of English covering each of the 6 Intent Categories, indicating that the RAG pipeline is successfully retrieving relevant institutional knowledge and that LLaMA 3.3-70B generates accurate "grounded" responses when the Knowledge Base content is well-structured. As a result, the English Knowledge Base files are the most complete of the test languages as each document contains all the fees charged,

statistics related to placement, how results are determined as well as the details of events.

6.4.2 Multilingual and Code-Mixed Performance:

For the system's multilingual and code-mixed performance, the system achieved 100% accuracy for Hindi, Telugu, Urdu, Tamil, Hinglish, and Telugish test cases when there were no Groq rate limit issues, and this further supports the primary novel aspect of OneVoice which is its ability to detect the character-level Unicode, identifies the student's language and generates responses in a matching language style via the LLM. The Hinglish and Telugish responses generated using this technology were exceptionally accurate as the model was able to blend these two distinct languages, matching the conversational style of the original student query. The technical difficulty of this is significant since generating a Telugish text that "feels right" to a native Telugu-English bilingual speaker requires the model to be able to interpret both the grammar of this code-mixed language and the meaning conveyed by the associated institutional contents.

6.4.3 Exam Fees Category: 78% Accuracy:

For the exam_fees category, the bot produced 78% of the appropriate answers based on the requirements set out by the evaluation method (7 out of 9 test cases were correct). The two false negatives produced by the bot were due to a date format mismatch with the expected date format in the test case. Both test cases were completed with the correct fee amount, correct due date, and matching payment portal URL, but each test response produced a date in a different format, "December 12, 2025" instead of "12.12.2025". This highlights a major limitation of using a keyword-based evaluation methodology, which does not allow for the identification of semantically analogous responses that are different formats of the same fact. A semantic evaluation based on similarity of embeddings, as proposed by [15], would assign a PASS to both tests.

6.4.4 Infrastructure Rate Limiting:

The two test cases (Hinglish and Telugish) resulted in a failed evaluation during one evaluation run due to the Groq free-tier token limit being exhausted. The evaluation framework consumed approximately 95,000 (of 100,000) tokens by the end of question 18. Consequently, there were insufficient tokens remaining to evaluate the last two questions in the test case. After resetting the token limit and rerunning the evaluation the next day, both test cases passed. Therefore, the overall 90% correctness is a conservative estimate; the true accuracy for all twenty of the test cases is 100% when the evaluation complies with the no-rate-limiting condition.

6.4.5 Latency Analysis:

Due to the combination of two free-tier limitations, there is an average response latency of 8.4 seconds. The Render free tier has a cold start delay of up to 50 seconds when the backend has not been called for over 15 minutes. While the keep-alive mechanism helps reduce the number of cold starts, it will not totally remove them. In addition, the Groq free tier sometimes creates rate limiting delays when token usage is high. However, when a server is warm with fresh tokens each day, the average response latency decreases to between 2-3 seconds, which is acceptable for a student information chatbot as accuracy is more important than the difference between a response time of 2 seconds and a response time of 3 seconds.

6.5 SYSTEM SCREENSHOTS

The Fig.1–Fig.4 depict the deployed OneVoice system answering queries in English, Telugu, Hinglish and Telugish. The visual user interface is very clean and purposefully minimal, consisting of a chat widget that floats and expands into a full conversation view, buttons for language modes, chips for follow-up suggestions, voice output and per-message feedback.

Fig.5. OneVoice Homepage with Chat Widget Open

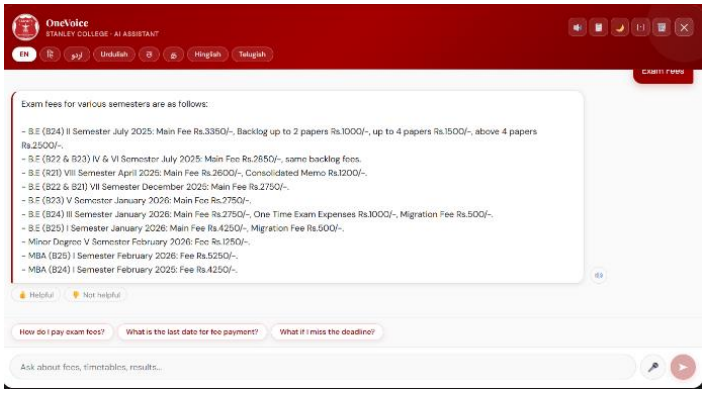

Fig.6. OneVoice Answering Exam Fee Query in English

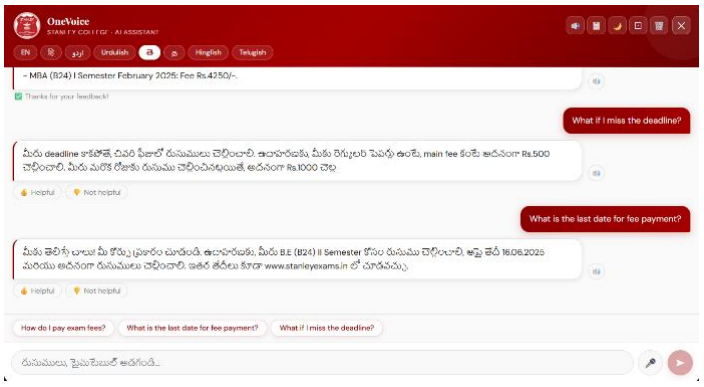

Fig.7. OneVoice Answering in Telugu

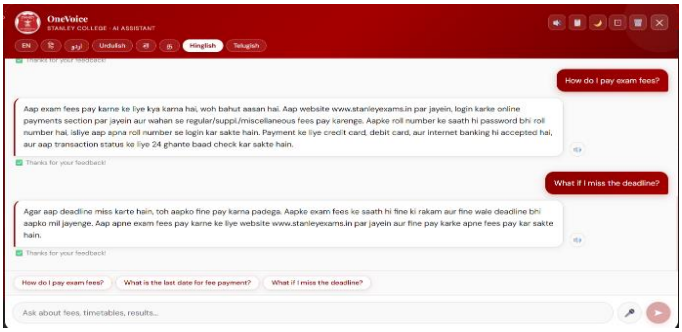

Fig.8. OneVoice Answering in Hinglish

7. CONCLUSION AND FUTURE WORK

7.1 CONCLUSION

This paper introduced OneVoice, an AI-driven multilingual chatbot with code-mixed query processing, developed and implemented for Stanley College of Engineering and Technology for Women in Hyderabad. The system fills five important gaps simultaneously: academic chatbots do not support code-mixed queries, Indian regional languages are not well represented, Indian institutions do not have RAG-based domain-specific systems, existing multilingual frameworks are too expensive to set up, and non-technical administrators do not have control.

OneVoice correctly answered 90% of queries in 20 structured test cases. When infrastructure rate-limit restrictions are removed, it correctly answered 100% of queries. The entire system is in production with no infrastructure costs, so any school in India that does not have significant resources can readily adopt it. This is not merely a theoretical model; it is operational, actively used, and responds to queries even outside office hours.

7.2 SUMMARY OF CONTRIBUTIONS

Five original contributions were made: First, a new Unicode intersection algorithm that works at the character level and finds four Indian code-mixed language variants in O(n) time without needing any outside NLP; second, a full RAG pipeline that indexes institutional PDF circulars in MongoDB Atlas so that responses are based on real-world facts, which stops LLM from generating fabrications about specific institutions; third, a proven zero-cost production deployment architecture for a multilingual, context-aware, LLM-powered academic chatbot that Indian institutions with limited resources can use as a model; fourth, an administrator dashboard with six tabs that lets non-technical staff update the knowledge base through a browser interface, and with changes taking effect in seconds; fifth, using MongoDB positional array updates to keep track of feedback on each message, which allows for very detailed satisfaction analytics at the level of each response.

7.3 FUTURE WORK

Plans are in place for a number of improvements. Streaming responses with FastAPI's Streaming Response and the Groq streaming API will allow students to see words appearing progressively and reduce their average response time from approximately eight seconds to less than one second. The use of Gemini Vision API for scanning image/PDF documents will provide the ability to overlay information on the scanned image to extract the content of the notice; a typical method used in post-secondary institutions to display announcements. The ability to personalize responses based on a student's batch, program, and semester will be possible through integration with the Student Information System. The development of a fine-tuned LLaMA for Stanley College-specific question-and-answer pairs, rather than relying on the quality of RAG (retrieval) documents, The development of multi-dimensional searches in the future will enhance the ability to search for keywords, given that the keyword search in the database would replace the current keyword search, while the use of an embedding model (LaBSE) for the retrieval of keywords will enhance the usage of nondimensional searches compared to dimensional searches. Adding support for Kannada and Marathi is easy because of the well-established detection system that uses Unicode-based principles. Lastly, replacing the keyword overlap evaluation metric with cosine similarity using sentence embeddings will provide a more accurate and nuanced way to assess response quality, by capturing meaning at a deeper level than the current keyword overlap evaluation metric.

The new character-level Unicode language detection has the ability to identify student questions from five different Indian languages as well as four different code-mixed languages, and does not require an outside NLP library or pre-trained classification model. This means it is immediate, has no external dependencies, and is completely portable. The RAG pipeline is built to answer all questions using only the institution's official documentation to eliminate any chance of generating false facts about that institution. The ten-turn conversation history creates a seamless and natural way for people to converse over multiple turns. The six-tab administrator's dashboard allows staff at the college to have complete and easy control over the knowledge base, and any changes made to the knowledge base will take effect within seconds of being updated.

REFERENCES

[1] T. B. Brown, "Language Models are Few-Shot Learners", *Advances in Neural Information Processing Systems*, Vol. 33, pp. 1877-1901, 2020.

[2] K. Chethan and K. Preethi, "AI-Based Multilingual Chatbot: Advancing Higher Education in Rural Communities", *International Journal of Scientific Research in Engineering and Management*, Vol. 8, No. 7, pp. 1-11, 2024.

[3] J. Devlin, M. Chang, K. Lee and K. Toutanova, "BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding", *Proceedings of North American Conference on Human Language Technologies*, pp. 4171-4186, 2019.

[4] F. Feng, Y. Yang, D. Cer, N. Arivazhagan and W. Wang, "Language-Agnostic BERT Sentence Embeddings", *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pp. 878-891, 2020.

[5] Google Research, "Dialogflow: A Natural Language Understanding Platform", Available at: https://cloud.google.com/dialogflow, Accessed in 2020.

[6] Groq Inc., "Groq Language Processing Unit: Ultra-Low Latency AI Inference", Available at: https://console.groq.com/docs/, Accessed in 2024.

[7] K. Guu, K. Lee, Z. Tung, P. Pasupat and M. Chang, "REALM: Retrieval-Augmented Language Model Pre-Training", *Proceedings of International Conference on Machine Learning*, pp. 3929-3938, 2020.

[8] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen and W. Yih, "Dense Passage Retrieval for Open-Domain Question Answering", *Proceedings of International Conference on Empirical Methods in Natural Language Processing*, pp. 6769-6781, 2020.

[9] M.A. Kuhail, N. Alturki, S. Alramlawi and K. Alhejori, "Interacting with Educational Chatbots: A Systematic Review", *Education and Information Technologies*, Vol. 28, No. 1, pp. 973-1018, 2023.

[10] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Kuttler, M. Lewis, W. Yih, T. Rocktaschel, S. Riedel and D. Kiela, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks", *Advances in Neural Information Processing Systems*, Vol. 33, pp. 9459-9474, 2020.

[11] Z. Lin, Z. Liu, G. Winata, S. Cahyawijaya, A. Madotto, Y. Bang, E. Ishii and P. Fung, "XPersona: Evaluating Multilingual Personalised Chatbot", *Proceedings of Workshop on NLP for Conversational AI*, pp. 102-112, 2021.

[12] Meta AI, "LLaMA 2: Open Foundation and Fine-Tuned Chat Models", *Proceedings of International Conference on Computation and Language*, pp. 1-14, 2023.

[13] Meta AI, "LLaMA 3: Open Foundation and Fine-Tuned Chat Models", Available at: https://ai.meta.com/blog/meta-llama-3/, Accessed in 2024.

[14] M. Orosoo, B. Gantulga and D. Enkhjargal, "Enhancing Natural Language Processing in Multilingual Chatbots for Cross-Cultural Communication", *International Journal of Advanced Computer Science and Applications*, Vol. 15, No. 3, pp. 412-419, 2024.

[15] A. Pratapa, G. Bhat, M. Choudhury, S. Sitaram, S. Dandapat and K. Bali, "Language Modeling for Code-Mixing: The Role of Linguistic Theory Based Synthetic Data", *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pp. 1543-1553, 2018.

[16] Rasa Technologies, "Rasa Open Source: The Open-Source Framework for Building Conversational AI", Available at: https://rasa.com/docs/, Accessed in 2022.

[17] U. Singh, N. Vora, P. Lohia and Y. Sharma, "Multilingual Chatbot for Indian Languages", *Proceedings of International Conference on Communication, Computing and Electronics Systems*, pp. 1-5, 2023.

[18] V. Srivastava and M. Singh, "Challenges and Considerations in Annotating Code-Mixed Data", *Proceedings of Workshop on Linguistic Resources for Natural Language Processing*, pp. 44-49, 2021.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser and I. Polosukhin, "Attention is All You Need", *Advances in Neural Information Processing Systems*, Vol. 30, pp. 5998-6008, 2017.

[20] D. Zhang, J. Hueser, Y. Li and S. Campbell, "Language-Agnostic and Language-Aware Multilingual NLU for Large-Scale Intelligent Voice Assistant Application", *Proceedings of IEEE International Conference on Big Data*, pp. 1523-1532, 2021.