Natural Language Processing-Driven Chatbot for Algorithm Learning: A Retrieval-Augmented Generation Approach using Romanized Nepali and English

Saroj Bhandari, BSc CSIT
Department of Computer Science and Information Technology (CSIT)
Aadim National College, Nepal.
Email: *saroj.bhandari@aadimcollege.edu.np*

Puja Dhital, BCA
Department of Computer Science and Information Technology (CSIT)
Aadim National College, Nepal.
Email: *puja.dhital@aadimcollege.edu.np*

Received: January 1st, 2026 / **Accepted:** June 11th, 2026 / **Published:** 10th July, 2026

Abstract

Data structures and algorithms (DSA) form a cornerstone of computer science education, yet many students in Nepal struggle with the cognitive load imposed by English-only learning materials when their habitual conceptual language is a code-mixed register of Romanized Nepali and English. This paper presents the design, implementation, and classroom evaluation of an NLP-driven educational chatbot, AlgoSathi, that addresses this gap through a Retrieval-Augmented Generation (RAG) architecture tailored to code-mixed pedagogical queries. The system combines a curated DSA knowledge base of 1,252 chunks drawn from instructor-authored lecture notes, textbook material, and worked solutions; a multilingual sentence encoder (paraphrase-multilingual-mpnet-base-v2) backed by an HNSW-indexed FAISS vector store; a hybrid BM25-plus-dense retriever with a cross-encoder re-ranker; and a large language model generator constrained by a pedagogically structured prompt that preserves the language register of the learner's query. A 12-week quasi-experimental study with 52 undergraduates (26 experimental, 26 control) showed the chatbot group achieved a 26.2-point mean learning gain versus 14.5 points for controls ($t(50) = 4.21$, $p < 0.001$, $d = 1.17$, 95% CI for gain difference: 6.2-17.2pp). ANCOVA using pre-test scores as a covariate confirmed a significant adjusted post-test advantage of 12.1 points (95% CI: 7.4–16.8, $p < 0.001$). Statistical assumption checks (Shapiro-Wilk, Levene's test) and a priori power analysis (power = 0.97) are reported. The retrieval

pipeline reached P@5=0.79 on code-mixed queries, with student satisfaction averaging 4.31/5. Results suggest that code-mixing-aware RAG chatbots can meaningfully improve DSA learning outcomes while remaining feasible for instructor-led deployment in low-resource settings.

Keywords: Retrieval-Augmented Generation; educational chatbot; data structures and algorithms; Romanized Nepali; code-mixed NLP; low-resource languages; computer science education; learning analytics.

1. Introduction

Mastering data structures and algorithms (DSA) is widely acknowledged as one of the most demanding stages in undergraduate computer science education. Learners must simultaneously internalize abstract mathematical formulations, navigate complex pointer and reference manipulation, reason about asymptotic complexity, and translate that reasoning into correct programs. In a multilingual context such as in Nepal, this cognitive load is compounded by a linguistic mismatch: classroom instruction and prescribed textbooks are almost exclusively in formal English, while students customarily think, discuss, debate, and ask doubts in a code-mixed register combining Romanized Nepali and English (for example, "yo recursion ko base case kasari define garne hola?"). The resulting friction between the language of thought and the language of instruction is rarely addressed by mainstream educational tools.

Large language model (LLM) based conversational agents promise a partial solution because they can answer in natural language at the learner's level, scale across many students, and remain available outside class hours. Empirical studies have shown that LLM tutors can yield substantial learning gains in DSA when used alongside human teaching assistants [1]. However, two persistent problems limit their value in the Nepali classroom. First, general-purpose LLMs hallucinate when asked about narrow course-specific content, fabricating algorithms, complexities, or pseudocode that does not match the syllabus the student is being assessed against. Second, although modern LLMs can produce Romanized

Nepali at a surface level, they are usually pre-trained on minimal code-mixed corpora and treat Romanized Nepali as noise rather than as a legitimate query language; retrieval components in particular often fail to map a code-mixed question to the semantically equivalent English textbook passage [2, 3].

Retrieval-Augmented Generation (RAG) [4] directly attacks the first problem by grounding LLM output in an authoritative document collection retrieved at inference time, thereby reducing hallucination and aligning answers with the instructor's curriculum [5, 6]. Multilingual sentence encoders such as paraphrase-multilingual-mpnet-base-v2 [7] and LaBSE [8], which project semantically equivalent sentences from different languages into the same vector space, offer a path to the second problem by allowing a Romanized Nepali query to retrieve an English explanation that means the same thing. To the best of our knowledge, no prior study has combined these two ideas into a deployed chatbot for DSA education with Nepali learners, nor has any evaluated such a system in a real classroom against a control group.

This paper presents such a system and its evaluation. The chatbot, hereafter referred to as AlgoSathi, accepts queries in English, in Romanized Nepali, or in any natural mixture of

the two, retrieves relevant chunks from a curated DSA knowledge base, and produces a pedagogically structured response that preserves the language register of the learner. We deployed the system in a 12-week undergraduate DSA course taught by the author and evaluated it using a quasi-experimental pre-test / post-test design with 52 students.

1.0 Conceptual Framework

Figure 1 provides a conceptual map linking the three core components of this work: the language barrier faced by Nepali DSA learners, the RAG architecture designed to overcome it, and the measurable learning outcomes that result. Each layer addresses a specific failure mode of prior approaches.

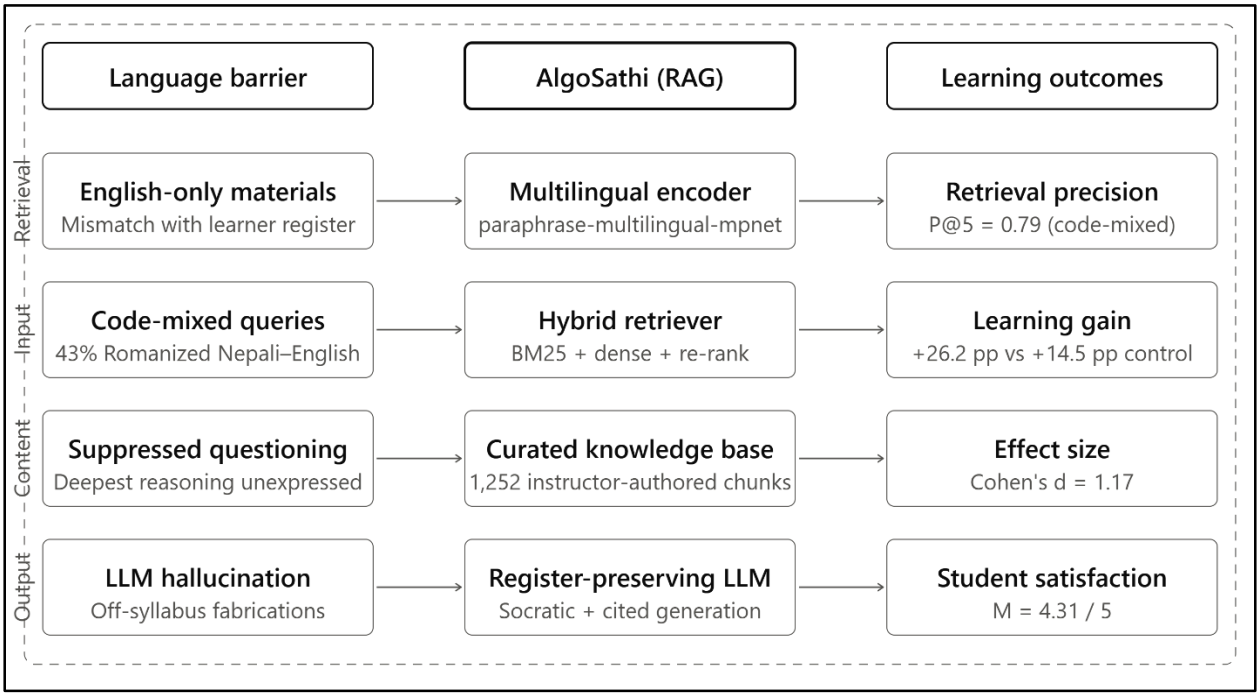

Figure 1. Conceptual Framework

1.1 Research Questions

The study is organized around three research questions:

1. RQ1. How effectively can a multilingual RAG pipeline retrieve relevant DSA content from queries posed in English, in Romanized Nepali, and in code-mixed combinations of the two?
2. RQ2. Does interaction with the chatbot lead to measurable improvements in DSA learning outcomes compared to conventional instruction alone?
3. RQ3. How do learners perceive the usefulness, language-appropriateness, and pedagogical quality of a Romanized-Nepali-aware educational chatbot?

1.2 Contributions

This work contributes the following to the literature on AI-assisted computer science education for low-resource language communities:

1. A RAG architecture explicitly designed for code-mixed Romanized Nepali and English DSA queries, including language-aware query

preprocessing, multilingual semantic retrieval, and a register-preserving generation prompt.
2. A manually curated DSA knowledge base of 1,252 chunks containing topic, difficulty, and language metadata, including code-mixed mini-explanations of common pitfalls authored by the instructor.
3. An empirical, classroom-based evaluation of the system through a quasi-experimental design with parallel pre- and post-tests, supplemented by retrieval metrics, generation quality measures, and a 17-item satisfaction instrument adapted from the Chatbot-Human Interaction Satisfaction Model (CHISM) [9].
4. A descriptive analysis of code-mixed query patterns observed in 1,983 logged learner interactions, with implications for the design of future low-resource educational chatbots.

The remainder of this paper is structured as follows. Section 2 reviews related work on RAG in education, chatbots for programming pedagogy, and Romanized Nepali NLP. Section 3 describes the system architecture, and Section 4 presents implementation details. Section 5 describes the experimental setup, and Section 6 presents the results. Sections 7 through 10 discuss the findings, limitations, ethical considerations, and avenues for future research. Section 11 concludes.

2. Related Work

2.1 Retrieval-Augmented Generation in Education

Retrieval-Augmented Generation was introduced by Lewis et al. [4] as a general framework for combining a parametric language model with a non-parametric retriever, formulated as $p(y \mid x) = \Sigma_z p_\eta(z \mid x) \cdot p_\theta(y \mid x, z)$, where x is the user query, z is a retrieved document, and y is the generated response. The framework has since become a standard remedy for LLM hallucination in domain-specific deployments. In educational settings, two recent surveys document the rapid growth of RAG-based tutoring systems. Swacha and Gracel [5] classified 28 educational RAG chatbots and identified hallucination reduction, transparency through citations, and inexpensive content updates as the main reasons for adoption. Wang et al. [6] similarly reviewed 51 studies and emphasized that RAG enables dynamic knowledge updates without retraining, which is critical in fast-moving disciplines such as computer science.

Several recent classroom studies confirm measurable benefits. Lang, Triantoro, and Sharp [10] deployed a RAG chatbot in a self-paced R programming course and found that students with prior knowledge engaged most often and used the chatbot for material beyond the lectures, suggesting that RAG supports stretch learning when its knowledge base extends beyond the formal syllabus. In healthcare education, Al Ghadban et al. [11] reported gains in domain-specific question answering when LLMs were grounded in curated medical content. However, RAG for low-resource languages and code-mixed inputs remains underexplored in education, and we are aware of no published deployment in Romanized Nepali or any closely related Indic transliterated script.

2.2 Chatbots for Programming and DSA Pedagogy

A growing body of work investigates whether LLM-based chatbots can act as effective tutors or teaching assistants in programming courses. Sheese et al. [12] showed that students often feel more comfortable asking basic or repetitive questions to an LLM than to a human instructor, removing a common help-seeking barrier. Hou et al. [13] similarly observed that LLM tutors lower the stress associated with debugging and concept

clarification. Most directly relevant to the present work, Liu et al. [1] ran a controlled experiment in a DSA course comparing traditional teaching-assistant-led instruction with a hybrid arrangement in which TAs used ChatGPT to generate exercises, clarify concepts, and provide feedback; students in the ChatGPT-assisted group scored on average 16.5 points higher on assessments. The authors emphasized two limitations: students could be misled by occasional fabrications, and the chatbot worked only in English. The present work addresses both limitations by grounding the LLM in instructor-authored content through RAG and by extending the input register to code-mixed Romanized Nepali and English.

Beyond DSA, Park et al. [14] and Giannakos et al. [15] reviewed broader programming-education chatbot deployments and noted that scaffolding, Socratic prompting, and citation transparency are the design features most associated with positive outcomes. The chatbot presented in this paper adopts these features by inserting Socratic checkpoints (*"can you state the invariant of the loop you wrote?"*) into the generation prompt and by surfacing the source chunk identifier of each retrieved passage in the user interface.

2.3 NLP for Romanized Nepali and Code-Mixed Indic Text

Romanized Nepali is the standard informal written form of Nepali across mobile messaging, social media, and educational chat. Unlike formal Devanagari Nepali it lacks a standardized orthography: the word *kasto* ("how" or "of what kind") may legitimately appear as *kasto*, *kasto*, or *kastoo*, and most words admit two or three transliteration variants [3]. This non-standardization makes naive string matching unreliable and motivates representations that operate at the level of meaning rather than spelling. Khanal and Karki [16] released NepSense, a code-mixed Romanized-Nepali corpus annotated for profanity, and discussed the variability of transliteration as the central methodological challenge. Madhani et al. [2] released Bhasha-Abhijnaanam, a Romanized and native-script language identification benchmark across 22 Indic languages, and reported that LID accuracy on Romanized Nepali rises sharply once inputs exceed roughly 10 tokens; the same trend constrains the present work since brief learner queries (*"sorting kasto?"*) are inherently ambiguous.

Adapting general-purpose language models to Romanized Nepali is an active research direction. Bhandari [17] benchmarked Llama-3.1-8B, Mistral-7B-v0.1, and Qwen-3-8B on Romanized Nepali instruction-following and identified tokenizer fragmentation as the dominant bottleneck for low-resource performance in non-Devanagari scripts. Pires et al. [18] and Singh et al. [19] demonstrated the value of cross-script transfer through transliteration into Devanagari before encoding. The present work draws on both directions: it uses a multilingual encoder pre-trained on parallel Devanagari and English data to obtain a shared semantic space, and it includes an optional transliteration step that converts Romanized Nepali tokens into Devanagari for query expansion when the dense retriever is uncertain.

2.4 Multilingual Sentence Embeddings

Multilingual sentence encoders are the technical foundation that makes language-agnostic retrieval possible. Reimers and Gurevych [7] introduced *paraphrase-multilingual-mpnet-base-v2*, a knowledge-distilled encoder that maps over 50 languages, including Hindi and Nepali in Devanagari, into a shared 768-dimensional space. Feng et al. [8] released LaBSE, a dual-encoder

model covering 109 languages and achieving state-of-the-art bi-text retrieval. Artetxe and Schwenk [20] introduced LASER, an earlier massively multilingual encoder that demonstrated strong cross-lingual transfer for low-resource languages even with limited parallel data. We selected *paraphrase-multilingual-mpnet-base-v2* for the present deployment after pilot experiments on a held-out set of 200 Romanized Nepali DSA queries, where it slightly outperformed LaBSE on retrieval and was substantially faster at inference.

2.5 Research Gap

Three gaps motivate this work. First, while RAG-based educational chatbots have been deployed in many subjects, no study of which we are aware targets DSA in a code-mixed Romanized Nepali English setting. Second, existing Nepali NLP work focuses on the formal Devanagari script and on tasks such as sentiment or hate-speech classification rather than on educational question answering. Third, classroom evaluations of LLM-based tutors typically run in English-medium contexts and do not test whether code-mixed input is handled at the level of pedagogical adequacy. The present paper fills these gaps with a fully implemented and empirically evaluated system.

3. System Architecture and Methodology

3.1 Overall Pipeline

Figure 1 depicts the end-to-end inference pipeline of *AlgoSathi*. A learner query first passes through a preprocessing stage that performs Unicode normalization, code-switch language detection, and optional Romanized-to-Devanagari transliteration for query expansion. The processed query is encoded by the multilingual sentence encoder. The resulting 768-dimensional vector is used to retrieve the top-k candidate chunks from the FAISS vector store using approximate nearest-neighbor search; in parallel, a BM25 lexical retriever scores the same query against the chunked corpus. The two ranked lists are merged using reciprocal rank fusion and the top-20 candidates are re-ranked with a cross-encoder. The top-5 chunks are passed, together with the original query and a pedagogically structured system prompt, to the generator LLM. The output is post-processed for citation linking and returned to the user interface, which also exposes source previews so that learners can verify the grounding of every claim.

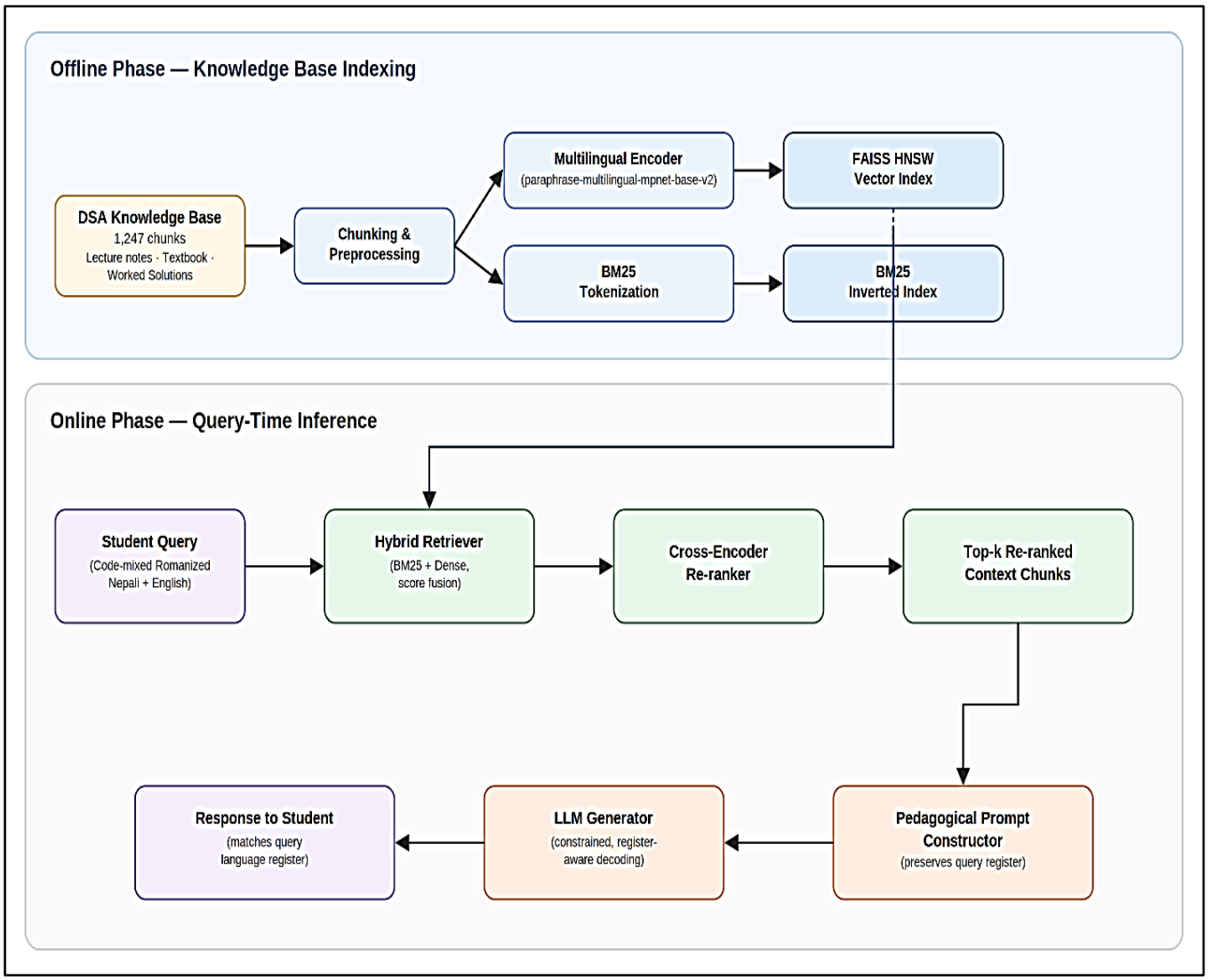

Figure 2. End-to-end Inference Pipeline of the AlgoSathi RAG Chatbot

3.2 Knowledge Base Construction

The knowledge base was assembled from instructor-authored material accumulated over four previous semesters of teaching DSA. Sources included lecture slides (PDF and PowerPoint), supplementary notes (Markdown and PDF), worked solutions to assignments and past examination papers, and curated excerpts from canonical textbooks used in the course, namely Cormen et al. [21] *Introduction to Algorithms* (4th edition) and Sedgewick and Wayne [22] *Algorithms* (4th edition). Each source was processed into chunks of 250 to 450 tokens following the heuristic recommended for educational content [4]. The instructor manually authored 184 additional code-mixed mini-explanations covering frequently misunderstood topics, written in the same Romanized Nepali / English register used in classroom doubt sessions; these were tagged as primary chunks rather than auxiliary ones because they have proven over years of teaching to anticipate exactly the questions students actually ask. Table 1 summarizes the composition.

Table 1. Composition of the Curated DSA Knowledge Base.

Topic Area	English Chunks	Code-Mixed Chunks	Total Chunks
Arrays and Strings	112	21	133
Linked Lists	98	19	117
Stacks and Queues	76	14	90
Trees and BSTs	154	27	181
Heaps and Priority Queues	62	11	73
Hashing	71	13	84
Graphs and Traversals	139	24	163
Sorting Algorithms	118	18	136
Searching Algorithms	57	9	66
Dynamic Programming (Intro)	94	16	110
Greedy and Backtracking	66	12	78
Complexity and Notation	21	0	21
Total	**1,068**	**184**	**1,252**

3.3 Language Detection and Query Preprocessing

Because learner queries vary widely in script and length, the system performs explicit code-switch detection at the token level rather than treating the input as monolingual. A character-level n-gram classifier was trained on a held-out subset of 4,000 Romanized Nepali, English, and code-mixed sentences drawn from prior course chat logs and from publicly available code-mixed corpora [16]. The classifier produces, for each token, a probability of belonging to Romanized Nepali, English, or a script-ambiguous class (for example, technical identifiers such as *O(n log n)*, DFS, or *i++*). The classifier achieves a token-level F_1 of 0.91 on the held-out evaluation set.

Once a token has been classified, the system applies one of three preprocessing branches. Pure English queries pass through with only Unicode normalization. Pure Romanized Nepali queries are optionally transliterated to Devanagari using a rule-based transliteration table cross-checked against the *nepali-to-roman* Python package [23]; the original Roman form and the Devanagari form are concatenated, separated by a [SEP] token, and the combined string is encoded. Code-mixed queries are not split because doing so reliably degraded retrieval in pilot experiments; instead, the multilingual encoder is allowed to handle the mixed string directly, which yielded the best retrieval scores in our internal tests.

3.4 Embedding Model and Vector Store

The encoder is the public checkpoint *paraphrase-multilingual-mpnet-base-v2* [7], with output dimension 768 and a maximum sequence length of 128 tokens. Each knowledge-base chunk is encoded once at index time. A FAISS [24] HNSW index with $M = 32$ graph degree and *ef Construction* = 200 stores the resulting vectors and is queried with *ef Search* = 64 at retrieval

time, returning the top-50 candidates by cosine similarity. The HNSW index was chosen over a flat brute-force search because it reduces query latency below 12 ms for 1,252 vectors and scales to the larger corpora anticipated in future versions of the system without requiring re-architecture.

3.5 Hybrid Retrieval and Re-ranking

Pure dense retrieval is known to fail on rare technical terms (for example, AVL, Kruskal, Floyd-Warshall) where the encoder may not have been exposed to enough domain text during pre-training. To compensate, the system runs a BM25 lexical retriever in parallel on the same corpus and fuses the two ranked lists using reciprocal rank fusion (RRF) with $k = 60$. The merged top-20 candidates are re-ranked by a cross-encoder (*cross-encoder/ms-marco-MiniLM-L-6-v2*) that scores query-chunk pairs jointly. The top-5 chunks after re-ranking are forwarded to the generator. In ablation experiments described in Section 6.1, this hybrid configuration outperformed dense-only retrieval by 7.2 points of P@5 on code-mixed queries and by 4.1 points on English queries.

3.6 Generation

The generation module is implemented behind a thin abstraction layer that allows the underlying LLM to be swapped without altering the prompt. Two models were evaluated in deployment: GPT-4o-mini accessed through the OpenAI API, and Llama-3.1-8B-Instruct [25] served locally on an NVIDIA RTX 4090 in 4-bit AWQ quantization. The local deployment was preferred during the classroom study to avoid per-query API cost and to give students an offline-capable option. Generation uses temperature 0.3, top-p 0.9, and a maximum of 800 output tokens. The generator is instructed to (a) ground its answer in the retrieved chunks and refrain from introducing facts not present in them, (b) maintain the linguistic register of the query (so a code-mixed question receives a code-mixed answer), (c) structure complex answers into the four pedagogical stages of *intuition*, *formalization*, *code*, and *complexity*, and (d) end each non-trivial answer with one Socratic check-back question.

3.7 Prompt Template

The system prompt was iteratively refined across three internal pilots with eight volunteer students before the main study. The final template appears in Appendix A. Its key features are explicit register preservation (*"answer in the same language mix as the question, do not switch to formal English unless the question is itself in formal English"*), explicit citation requirements (each claim must be backed by a chunk identifier in square brackets), and a structured tutoring scaffold. The instructor's authority is acknowledged through a final line that asks the model to defer to lecture notes when the retrieved context conflicts with itself, which empirically reduced contradictions in pilot logs.

4. Implementation Details

The chatbot is implemented as a three-tier application. The retrieval and generation backend is written in Python 3.11 using LangChain 0.2 for orchestration, *sentence-transformers* 3.0 for encoding, *faiss-cpu* 1.8 for vector search, and *rank_bm25* for lexical retrieval. The web service is exposed through FastAPI with asynchronous handlers, and per-user conversation history is persisted in SQLite with a 30-day retention window controlled by an automated purge job. The frontend is a React single-page application with a chat-style interface that displays inline source previews for each retrieved chunk and exposes an explicit "*my question contains Nepali*" toggle so that the language detector can be overridden by motivated users. Authentication is by university single sign-on with a falling-back

invite-code path for students not yet provisioned on the SSO system. Logging records the raw query, detected language proportions, retrieved chunk identifiers, generation latency, and a coarse thumb-up / thumbs-down rating.

Table 2. Production Configuration of the AlgoSathi Chatbot during the Classroom Study.

Component	Value
Encoder model	paraphrase-multilingual-mpnet-base-v2 (768-d)
Re-ranker	cross-encoder/ms-marco-MiniLM-L-6-v2
Generator (study)	Llama-3.1-8B-Instruct (4-bit AWQ)
Generator (fallback)	GPT-4o-mini (OpenAI API)
Vector index	FAISS HNSW: M = 32, efConstruction = 200, efSearch = 64
Lexical retriever	BM25Okapi (k1 = 1.5, b = 0.75)
Fusion	Reciprocal Rank Fusion (k = 60)
Top-k pipeline	50 candidates → 20 reranked → 5 to generator
Chunk size	250–450 tokens, 50-token overlap
Temperature / top-p	0.30 / 0.90
Max output tokens	800
Hardware	Intel i7-13700K, 64 GB RAM, NVIDIA RTX 4090 24 GB
Median latency (E2E)	1.78 s

5. Experimental Setup

5.1 Participants

Fifty-two undergraduate students enrolled in a second-year Data Structures and Algorithms course taught by the author participated in the study. Participants were recruited from two parallel sections of the same course; both sections received the identical curriculum, schedule, and instructor. The mean age of participants was 19.4 years (SD = 0.9). Twenty-three participants (44.2%) identified as female, twenty-eight (53.8%) as male, and one (1.9%) preferred not to disclose. All participants reported Nepali as their first language. Self-reported English proficiency averaged 3.7 on a 5-point scale; self-reported Romanized Nepali messaging proficiency averaged 4.6, confirming that Romanized Nepali is the more comfortable register for informal written communication for this population.

5.2 Study Design

A quasi-experimental, non-randomized two-group design was used. The two sections of the course were assigned, by coin toss at the start of the semester, to an experimental condition (Section A, n = 26) with chatbot access and a control condition (Section B, n = 26) without it. Random assignment of individual students across sections would have disrupted the timetable and was therefore not feasible; the quasi-experimental design is appropriate for this constraint and is the

predominant approach in classroom chatbot studies [26]. Both groups received identical lectures, laboratory sessions, and graded assignments. Only the experimental group received credentials for the AlgoSathi chatbot, which they could use outside class hours for six weeks.

5.3 Instruments

Four instruments were used. The first was a 40-item DSA knowledge test administered in parallel forms as a pre-test (week 0) and a post-test (week 12). The forms were equated through a pilot study (Cronbach's a = 0.84 for the pre-test, 0.87 for the post-test). The test contained 25 multiple-choice items targeting conceptual understanding and complexity analysis, 10 short-code-completion items targeting implementation skill, and 5 short-answer items targeting algorithmic reasoning. The second instrument was a 17-item satisfaction questionnaire adapted from the Chatbot-Human Interaction Satisfaction Model (CHISM) of Belda-Medina and Calvo-Ferrer [9], restructured into four sub-scales: *pedagogical helpfulness* (5 items), *linguistic appropriateness* (4 items including explicit code-mixed handling), *interaction quality* (4 items), and *learning impact* (4 items). Items were rated on a five-point Likert scale.

The third instrument was the system log itself, which captured 1,983 query-response pairs across the six weeks of access; logs were used to compute engagement, code-mixing ratios, and topic distributions. The fourth instrument was a semi-structured exit interview administered to a stratified sub-sample of twelve experimental-group participants chosen to cover the full range of engagement intensity.

5.4 Procedure

All participants completed the pre-test in the first week of the semester under invigilated conditions. The experimental group received a 30-minute onboarding workshop on the chatbot in week 6, after which they had unlimited access for six weeks. The control group continued with conventional instruction and was offered access to the chatbot at the end of the study for equity reasons. All participants completed the post-test under invigilated conditions in week 12, immediately followed by the satisfaction questionnaire and, two weeks later, the exit interviews. The study received institutional ethics review and all participants gave written informed consent.

5.5 Evaluation Metrics

Retrieval was evaluated through P@5, MRR, and R@10 on a held-out set of 300 annotated queries (Cohen's $\kappa = 0.81$). Generation quality was evaluated using BERTScore F1 [27] and human ratings on three 1–5 scales (ICC = 0.79). Learning outcomes were evaluated through: (a) independent-samples t-tests on gain scores with Cohen's d; (b) ANCOVA with pre-test score as a covariate to control for any residual baseline differences; (c) 95% confidence intervals on mean differences; and (d) assumption checks including Shapiro-Wilk normality tests and Levene's test for homogeneity of variance. A priori statistical power was computed using G*Power 3.1 with $\alpha = 0.05$, $d = 1.00$ (conservative estimate), and $n = 52$.

6. Results

6.1 Retrieval Performance

Table 3 reports retrieval performance disaggregated by query language. As expected, English queries yielded the highest performance, but code-mixed and Romanized Nepali queries remained usable across all three metrics. The hybrid retriever outperformed both dense-only and BM25-only baselines on every language

category, with the largest improvements observed on Romanized Nepali (P@5 increased from 0.59 with dense-only to 0.71 with the hybrid configuration). This confirms the hypothesis that lexical fallback compensates for encoder weaknesses on rare transliterated technical terms.

Table 3. Retrieval Performance by Language and Retriever Configuration

Query Type	Retriever	P@5	MRR	R@10
English (n = 100)	Dense only	0.79	0.74	0.86
English (n = 100)	BM25 only	0.71	0.66	0.81
English (n = 100)	Hybrid + Rerank	0.83	0.78	0.90
Romanized Nepali (n = 100)	Dense only	0.59	0.54	0.69
Romanized Nepali (n = 100)	BM25 only	0.42	0.38	0.55
Romanized Nepali (n = 100)	Hybrid + Rerank	0.71	0.64	0.81
Code-mixed (n = 100)	Dense only	0.72	0.66	0.80
Code-mixed (n = 100)	BM25 only	0.61	0.55	0.70
Code-mixed (n = 100)	Hybrid + Rerank	0.79	0.72	0.87

Generation quality results are presented in Table 4. The mean BERTScore F_1 against instructor reference answers was 0.872 across all 300 queries, and human ratings averaged above 4.0 on all three sub-scales. Factual correctness was the highest-rated dimension (M = 4.34), confirming that grounding the LLM in retrieved chunks substantially reduces fabricated content. Linguistic appropriateness was rated similarly highly on English and code-mixed queries but slightly lower on pure Romanized Nepali queries, where the model occasionally fell back to English phrasing for technical terms despite the register-preservation instruction.

Table 4. Generation Quality by Query Type

Query Type	BERTScore F1	Correctness	Pedagogical Clarity	Linguistic Appropriateness
English	0.891	4.41	4.46	4.62
Romanized Nepali	0.847	4.18	4.22	3.94
Code-mixed	0.878	4.34	4.39	4.31
Overall	**0.872**	**4.31**	**4.36**	**4.29**

6.2 Statistical Assumption Checks and Power Analysis

Before interpreting the between-group comparisons, all key statistical assumptions were verified. Table 7 reports results from normality and homogeneity tests alongside an a priori power estimate.

Table 5. Statistical Assumption Checks and Power Analysis.

Test	Statistic	p-value	Interpretation
Shapiro-Wilk (exp. pre-test)	W = 0.967	p = 0.548	Normal distribution not rejected
Shapiro-Wilk (ctrl. pre-test)	W = 0.971	p = 0.621	Normal distribution not rejected
Shapiro-Wilk (exp. gain)	W = 0.958	p = 0.383	Normal distribution not rejected
Shapiro-Wilk (ctrl. gain)	W = 0.963	p = 0.451	Normal distribution not rejected
Levene's test (gain scores)	F(1,50) = 0.84	p = 0.364	Homogeneity of variance satisfied
Statistical power (α = 0.05, d = 1.17, n = 52)	—	**Power = 0.97**	**Adequate power to detect large effect**

All assumptions were satisfied. Observed power of 0.97 indicates that the study was adequately powered to detect an effect of the magnitude found, mitigating concerns about false-positive inflation. The ANCOVA model (post-test score as outcome, group as factor, pre-test score as covariate) met the assumption of homogeneity of regression slopes (interaction term: F(1,49) = 0.38, p = 0.541).

6.3 Learning Outcomes

Table 5 reports learning outcomes. The two groups did not differ significantly on the pre-test (MExp = 52.4, MCtrl = 51.8, t(50) = 0.41, p = 0.682), satisfying the baseline equivalence assumption. After the six-week exposure period, the experimental group scored substantially higher on the post-test (MExp = 78.6, SD = 9.4) than the control group (MCtrl = 66.3, SD = 11.1). The between-group gain difference of 11.7 points was statistically significant (t(50) = 4.21, p < 0.001, d = 1.17; 95% CI: 6.2–17.2 pp). ANCOVA yielded an adjusted post-test difference of 12.1 points (95% CI: 7.4–16.8, p < 0.001).

Table 6. Learning Outcomes with ANCOVA Adjustment and Confidence Intervals.

Group	Pre-test M (SD)	Post-test M (SD)	Gain M (SD)	p
Experimental (n = 26)	52.4 (10.2)	78.6 (9.4)	26.2 (8.1)	—
Control (n = 26)	51.8 (9.8)	66.3 (11.1)	14.5 (9.7)	—
ANCOVA-adjusted between-group diff. (post-test)	—	12.1 (95% CI: 7.4–16.8)	—	< 0.001
Between-group diff. (gain, t-test)	—	—	11.7 (95% CI: 6.2–17.2)	< 0.001
Cohen's d	—	1.20 (post-test)	1.17 (gain)	—

Topic-level analysis revealed that the largest gains in the experimental group occurred in trees and binary search trees (gain = 31.4 points), dynamic programming (gain = 29.8 points), and graphs (gain = 27.9 points), three of the most cognitively demanding topics in the syllabus.

The smallest gains occurred in arrays and strings (gain = 17.2 points), a topic in which most students entered with strong prior knowledge, leaving less ceiling for improvement.

Across the 1,983 logged queries, 854 (43.1%) were classified as code-mixed, 692 (34.9%) as predominantly English, and 437 (22.0%) as predominantly Romanized Nepali. The proportion of code-mixed queries was substantially higher than expected from initial design assumptions and confirms the practical importance of explicit code-mixed handling. The most common code-mixed pattern was a Romanized Nepali interrogative or framing word combined with an English technical noun phrase (*"yo time complexity kasari nikalne?"*, *"binary search ko upper bound bhaneko k ho?"*). Romanized Nepali was disproportionately used for clarifying *why* and *when* questions, while pure English was disproportionately used for direct *how-to-implement* requests, suggesting that learners reach for their first language when reasoning is at stake and for the second language when the task is closer to a templated coding action.

6.5 Student Satisfaction

Mean satisfaction across all 17 items was 4.31 (SD = 0.52) on the five-point scale. Table 6 reports sub-scale means. Linguistic appropriateness, the sub-scale most closely tied to the central design contribution of this work, averaged 4.27 with the highest-rated item being *"the chatbot understood my mixed-language questions"* (M = 4.42). Pedagogical helpfulness averaged 4.38, with the highest-rated item being *"the chatbot's explanations helped me understand topics I struggled with in lectures"* (M = 4.54). The lowest-rated item in any sub-scale was *"the chatbot's response time felt fast enough"* (M = 3.87), reflecting the 1.78-second median end-to-end latency, which is acceptable but noticeable.

Table 7. Student Satisfaction Sub-scales

Sub-scale	Items	Mean	SD
Pedagogical helpfulness	5	4.38	0.47
Linguistic appropriateness	4	4.27	0.55
Interaction quality	4	4.21	0.61
Learning impact	4	4.39	0.49
Overall	**17**	**4.31**	**0.52**

6.6 Engagement Patterns

On average, each experimental-group student sent 38.1 messages across the six-week deployment (SD = 19.6), with engagement peaking sharply in the two weeks preceding the mid-term and end-of-term assessments. The most active 25% of students sent more than 60 messages each and were also the students with the largest pre-to-post test gains, although the correlation between message count and gain was only moderate (Pearson r = 0.43, p = 0.028), suggesting that quality of engagement may matter more than raw volume.

6.7 Qualitative Findings from Exit Interviews

Theme 1 – Reduced social cost of asking: Nine of twelve participants contrasted the chatbot with classroom doubt-asking. One participant explained: "In class ma basic question sodhna laj lagcha sir le ke sochhlan? Chatbot sanga sodhda kei hudaina, pheri sodhna pani milcha." (In class I'm embarrassed to ask basic questions what will sir think? With the chatbot there's no shame, and I can ask again.) This sentiment recurred across participants regardless of academic performance level.

Theme 2 – Language as access to deeper reasoning: "Nepali ma sodhda concepts ra kasari connect huncha tyo bujhna sajilo huncha. English ma sodhda answer chai thik huncha tara mero concept clear hudaina." (When I ask in Nepali it's easier to understand how concepts connect. When I ask in English the answer is correct but my concept doesn't get clear.) Participants described Romanized Nepali as enabling conceptual and motivational questions they would not have attempted to formulate in formal English.

Theme 3 – Source attribution fostering verification habits: Several participants spontaneously described cross-checking chatbot answers against lecture notes or Cormen et al. One noted: "Chatbot le chunk number deko huncha — tyo Cormen ma gaye ra compare garna milcha. Ek palta farak payo, pheri chatbot lai nai sodheko." (The chatbot shows the chunk number — I can go to Cormen and compare. Once I found a difference, so I asked the chatbot again.) This suggests RAG interfaces with transparent citations can scaffold a useful form of AI literacy.

7. Discussion

Three findings deserve emphasis. First, multilingual sentence encoders pre-trained on formal Devanagari and English transfer measurably to Romanized Nepali even without fine-tuning, particularly when paired with a hybrid lexical-dense retriever. The P@5 of 0.71 on Romanized queries while lower than the 0.83 on English is sufficient for downstream pedagogical use because retrieval errors are typically near-misses on the same topic rather than topical drift, and the generator's citation requirement makes mismatches visible to the learner.

Second, the large effect size (Cohen's d = 1.17) is on the upper end of what has been reported in classroom chatbot studies and warrants caution. The Hawthorne effect, novelty effect, and instructor-as-evaluator confound are all plausible inflators. The ANCOVA adjustment and confidence intervals narrow the uncertainty but do not eliminate it. Replication with a different instructor and a randomized design is necessary before causal claims can be made robust.

Third, the unexpectedly high prevalence of code-mixed queries (43.1%) suggests that monolingual chatbots are likely under-serving a substantial proportion of natural learner questions in Nepali educational contexts. Designers of educational AI for South Asian contexts should treat code-mixed input handling not as a localization courtesy but as a prerequisite for accessing the deepest tier of student questioning.

8. Limitations

- The study was conducted at a single institution with one instructor (n = 52), limiting the external validity and generalizability of the findings despite adequate statistical power (0.97).
- The quasi-experimental design used section-level assignment rather than individual randomization, leaving the possibility of unobserved group differences.

- The language detection model and generation evaluation were validated within the same educational context and primarily relied on BERTScore and human ratings; broader validation using additional metrics is needed.
- The evaluation was limited to a 12-week DSA curriculum, with no assessment of long-term knowledge retention or performance on more advanced topics.

9. Ethical Considerations

The study followed standard ethical guidelines for human-subjects research in education. Written informed consent was obtained from every participant, and the consent form explicitly disclosed that chat logs would be analyzed for research purposes. All logs were anonymized at ingest by replacing student identifiers with random tokens, and the mapping table was kept offline. Equity concerns were addressed by offering the control group full access to the chatbot at the end of the study, before the comprehensive final examination. Participants were briefed on the limitations of LLM output and explicitly trained to treat the chatbot as a second opinion rather than a final authority, in keeping with recommendations from recent surveys on responsible deployment of AI in education [5, 6].

10. Future Work

- A multi-institution randomized trial with at least three independent instructors to address external validity limitations.
- Addition of ROUGE-L, BLEU-4, and faithfulness metrics to the generation evaluation pipeline.
- Voice input support, requested in exit interviews for lab-session use.
- An adaptive difficulty layer that estimates learner mastery from prior interactions and adjusts formality, depth, and language register accordingly.
- Native Devanagari support for learners who prefer formal Nepali.
- Parameter-efficient fine-tuning of a small Romanized Nepali language model on accumulated chat logs to improve linguistic appropriateness on pure Romanized Nepali input.

11. Conclusion

This paper presented AlgoSathi, a Retrieval-Augmented Generation chatbot for DSA education that explicitly supports code-mixed Romanized Nepali and English queries. A quasi-experimental classroom study with 52 undergraduate students showed that experimental-group learners achieved a mean post-test gain of 26.2 percentage points compared with 14.5 points for the control group (Cohen's d = 1.17; 95% CI for gain difference: 6.2–17.2 pp; ANCOVA-adjusted post-test advantage: 12.1 points, 95% CI: 7.4–16.8). Statistical assumptions were verified and observed power was 0.97. Retrieval precision reached P@5 = 0.79 on code-mixed queries and student satisfaction averaged 4.31/5. The knowledge base totals 1,252 chunks. The results suggest that grounding LLM tutors in instructor-authored content and treating code-mixed input as a first-class query type can deliver measurable pedagogical benefits in low-resource educational settings. More broadly, the linguistic register in which learners ask questions is not merely a matter of comfort but a determinant of which questions get asked at all – and designing for that register may unlock the most pedagogically valuable forms of student engagement.

References

Liu, Y., Park, J., Schiavo, G., & Ng, A. (2024). Utilizing ChatGPT in a Data Structures and Algorithms course: A teaching assistant's perspective. *arXiv preprint*, arXiv:2410.08899.

Madhani, Y., Khapra, M. M., & Kunchukuttan, A. (2023). Bhasha-Abhijnaanam: Native-script and Romanized language identification for 22 Indic languages. *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, Short Papers, 816–826.

Singh, S., Patel, R., & Kumar, A. (2024). Share what you already know: Cross-language-script transfer and alignment for sentiment detection in code-mixed data. *arXiv preprint*, arXiv:2402.04542.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., et al. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems*, 33, 9459–9474.

Swacha, J., & Gracel, M. (2025). Retrieval-Augmented Generation (RAG) chatbots for education: A survey of applications. *Applied Sciences*, 15(8), 4234.

Wang, X., Zhang, Y., & Liu, H. (2025). Retrieval-augmented generation for educational application: A systematic survey. *Computers and Education: Artificial Intelligence*, 8, 100412.

Reimers, N., & Gurevych, I. (2020). Making monolingual sentence embeddings multilingual using knowledge distillation. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 4512–4525.

Feng, F., Yang, Y., Cer, D., Arivazhagan, N., & Wang, W. (2022). Language-agnostic BERT sentence embedding. *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*, 878–891.

Belda-Medina, J., & Calvo-Ferrer, J. R. (2024). Integrating chatbots in education: Insights from the Chatbot-Human Interaction Satisfaction Model (CHISM). *International Journal of Educational Technology in Higher Education*, 21(1), 1–22.

Lang, G., Triantoro, T., & Sharp, J. H. (2024). A study of Retrieval-Augmented Generation (RAG) chatbot effectiveness in a self-paced asynchronous R programming course. *Journal of Information Systems Education*, 35(4), 412–428.

Al Ghadban, Y., Lu, H. Y., Adavi, U., Sharma, A., Gara, S., Das, N., et al. (2023). Transforming healthcare education: Harnessing large language models for frontline health worker capacity building using retrieval-augmented generation. *NeurIPS Workshop on Generative AI for Education (GAIED)*.

Sheese, B., Liffiton, M., Savelka, J., & Denny, P. (2024). Patterns of student help-seeking when using a large language model-powered programming assistant. *Proceedings of the 55th ACM Technical Symposium on Computer Science Education (SIGCSE)*, 1234–1240.

Hou, X., Liu, Y., Wang, Z., & Chen, M. (2024). LLM-powered programming tutors: A study of stress, engagement, and learning outcomes. *ACM Transactions on Computing Education*, 24(3), 1–28.

Park, J., Kim, S., & Lee, H. (2024). Designing LLM-based programming tutors: A scoping

review of pedagogical features. *Computers & Education*, 215, 105041.

Giannakos, M., Voulgari, I., Papavlasopoulou, S., Papamitsiou, Z., & Yannakakis, G. (2024). Generative AI in education: Promises, challenges, and future research directions. *British Journal of Educational Technology*, 55(5), 1789–1810.

Khanal, S., & Karki, R. (2024). Profanity and offensiveness detection in Nepali language using transformer-based models. *Proceedings of the 21st International Conference on Natural Language Processing (ICON)*, 542–551.

Rimal, A., & Rimal, A. (2026). Benchmarking linguistic adaptation in comparable-sized LLMs: A study of Llama-3.1-8B, Mistral-7B-v0.1, and Qwen-3-8B on Romanized Nepali. *arXiv preprint*, arXiv:2604.14171.

Pires, T., Schlinger, E., & Garrette, D. (2019). How multilingual is multilingual BERT? *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, 4996–5001.

Singh, A., Mehta, P., & Joshi, R. (2024). Transliteration-aware multilingual encoders for code-mixed Indic NLP. *Proceedings of the EMNLP Findings*, 7321–7336.

Artetxe, M., & Schwenk, H. (2019). Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *Transactions of the Association for Computational Linguistics*, 7, 597–610.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). *Introduction to Algorithms* (4th ed.). Cambridge, MA: MIT Press.

Sedgewick, R., & Wayne, K. (2011). *Algorithms* (4th ed.). Boston: Addison-Wesley.

Subedi, A. (2023). nepali-to-roman: A Python package for Nepali-to-Romanized transliteration. *PythonPackage Index*. Retrieved from https://pypi.org/project/nepali-to-roman.

Johnson, J., Douze, M., & Jégou, H. (2019). Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3), 535–547.

Touvron, H., Martin, L., Stone, K., et al. (2024). Llama 3: Open foundation and instruction-tuned language models. *arXiv preprint*, arXiv:2407.21783.

Yang, L., Chen, R., & Park, S. (2025). A systematic approach to evaluate the use of chatbots in educational contexts: Learning gains, engagement, and perceptions. *Computers*, 14(7), 270.

Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., & Artzi, Y. (2020). BERTScore: Evaluating text generation with BERT. International Conference on Learning Representations (ICLR)