AinnAssist

Your Legal Assistant for Bangladeshi Law

Susmit Talukder
Department of Electrical and Computer Engineering North South University Dhaka, Bangladesh
susmit.talukder@northsouth.edu

S. M Karimul Hassan
Department of Electrical and Computer Engineering North South University Dhaka, Bangladesh
karimul.hassan@northsouth.edu

Ornab Barua
Department of Electrical and Computer Engineering North South University Dhaka, Bangladesh
ornab.barua@northsouth.edu

Md Shafiq Rahman Nirzon
Department of Electrical and Computer Engineering North South University Dhaka, Bangladesh
shafiq.nirzon@northsouth.edu

***Abstract*—Accessing reliable legal information in Bangladesh remains a significant challenge for citizens, students, lawyers, and journalists due to the complexity of legal documents and language barriers. This paper introduces *AinnAssist*, an AI-powered legal assistant designed to democratize access to Bangladeshi laws, including the Constitution and Penal Code, through a user-friendly Telegram bot. The system is built on a Retrieval-Augmented Generation (RAG) framework that processes official government documents, applies preprocessing to ensure clean and structured text, and stores semantic embeddings in a FAISS vector database for efficient retrieval. Query handling is enhanced through reranking techniques and multilingual support, enabling interactions in both Bangla and English. AinnAssist further supports multimodal inputs, allowing users to submit text, voice, or images, which are processed using speech recognition and image captioning. A FastAPI backend integrates with a state-of-the-art large language model to generate context-aware, cited responses accompanied by confidence scores. By combining advanced natural language processing with accessible interfaces, AinnAssist empowers users with instant, accurate, and trustworthy legal guidance. This framework demonstrates how AI can bridge legal knowledge gaps in developing contexts, promoting inclusivity, informed citizenship, and justice accessibility.**

Keywords—AI legal assistant, Bangladesh law, Retrieval-Augmented Generation (RAG), Multilingual support, Telegram bot, Legal information access

I. INTRODUCTION

Access to legal information in Bangladesh remains a major barrier for citizens, students, journalists, and even legal practitioners. The challenge is compounded by complex legal jargon, low literacy rates, and the limited availability of affordable legal services [1]. Foundational legal documents, such as the Constitution of the People's Republic of Bangladesh and the Penal Code of 1860, are often difficult to navigate due to their dense structure and the absence of intuitive tools for searching or understanding them [2]. These difficulties are particularly pronounced in rural and marginalized communities, where citizens face obstacles in interpreting their rights, statutory laws, and constitutional provisions [3]. Existing resources, including printed handbooks and government websites such as bdlaws.minlaw.gov.bd, are static and non-interactive, lacking multilingual support and failing to accommodate diverse input modalities.

Recent advances in artificial intelligence (AI) have opened new possibilities for democratizing access to justice. Retrieval-Augmented Generation (RAG) frameworks, large language models (LLMs), and vector search techniques enable natural language interaction with complex legal corpora [4], [5]. AI-powered legal assistants can bridge the gap between dense legal texts and end-users by retrieving relevant information from trusted sources and generating accessible, context-aware responses. While several prototype systems exist in Bangladesh, such as the Bangladesh Legal RAG Assistant and Noyona Legal Assistant, they remain limited in scope. Current implementations are restricted to static documents, lack support for multimodal inputs (e.g., voice and image), and provide insufficient Bangla language processing [6], [7].

To address these challenges, this paper presents AinnAssist, a multimodal AI legal assistant designed to provide accurate, accessible, and trustworthy guidance on Bangladeshi laws through a user-friendly Telegram interface. The system supports interactions in Bangla and English, accepts queries via text, speech, and images, and delivers responses enriched with source citations and confidence scores. Unlike existing systems, AinnAssist emphasizes inclusivity through multilingual capabilities, context tracking for follow-up queries, and multimodal input handling.

The primary objectives of this work are threefold: (i) to enhance legal literacy by simplifying access to key legal documents; (ii) to empower underserved populations through multimodal and multilingual support; and (iii) to overcome the shortcomings of prior systems by introducing real-time processing, error-handling mechanisms, and an extensible architecture for future legal domains such as cyber law and labor law. Evaluation focuses on retrieval precision,

confidence scoring, and user feedback, demonstrating improved usability and reliability compared to existing baselines.

II. Related Work

Access to structured and comprehensible legal information in Bangladesh has long posed a significant challenge. The inherent complexity of legal documents, coupled with limited availability of affordable legal services, has restricted citizens' ability to understand and exercise their legal rights. Over the years, several initiatives have attempted to leverage artificial intelligence (AI) to improve legal literacy and accessibility; however, these efforts remain constrained in terms of scope, adaptability, and user accessibility.

A. *Bangladesh Legal RAG Assistant [8]*

The Bangladesh Legal RAG Assistant, developed by Risad-Raihan, represents one of the earliest AI-driven efforts aimed at legal information retrieval in Bangladesh. This system employs a Retrieval-Augmented Generation (RAG) pipeline, where FAISS is used as the vector storage mechanism and Gemini large language models (LLMs) generate responses based on queries. The system processes legal content stored in PDFs of Bangladeshi laws, allowing users to obtain answers by retrieving relevant sections from these documents.

Despite its innovative approach, the system has notable limitations. Its reliance on static PDF documents prevents it from reflecting ongoing legal amendments or incorporating real-time updates from official sources. Furthermore, it lacks multimodal input capabilities, meaning it can only handle text queries, which restricts accessibility for users who might prefer voice or image-based inputs. Additionally, the system does not provide a fully developed user interface, limiting its deployment for general public use. Consequently, while the Bangladesh Legal RAG Assistant demonstrated the potential of RAG pipelines in the legal domain, its practical applicability remained restricted.

B. *Noyona Legal Assistant [9]*

Noyona Legal Assistant, developed by sayeedk06 during a UNDP-supported SDG Hackathon, sought to address a pressing social need: providing women with access to legal assistance related to emergency situations and safety. Built using Dialogflow, the system allows users to interact with predefined intents and responses, offering guidance and emergency hotline information.

While Noyona addressed a critical societal issue, its functionality is limited to specific, predefined scenarios. It does not support dynamic retrieval of legal information from updated legal texts, nor can it handle broader constitutional or statutory questions. The restricted scope and lack of adaptability highlight the challenges in developing AI systems that can offer generalized legal assistance in a dynamic legal environment.

C. *LegalMate-AI [10]*

LegalMate-AI, developed by zunaidhasan, explored the integration of OpenAI GPT-based models to provide conversational legal support. Unlike previous systems, LegalMate-AI aimed for a more generalized legal assistant capable of addressing a broader range of user queries. However, it did not adopt vector-based retrieval methods such as FAISS, which limited the precision and relevance of the responses generated. Moreover, the system lacks robust support for the Bangla language, which is essential for accessibility in Bangladesh, as a significant portion of users may not be proficient in English.

D. *LegalAdvisorLLM [11]*

LegalAdvisorLLM, developed by izammohammed, serves as a proof-of-concept chatbot that leverages large language models to answer legal queries. Although it demonstrates the potential of AI in legal assistance, the system is not specifically tailored to Bangladeshi legal content. It lacks a retrieval-based pipeline, multimodal input handling, multilingual capabilities, and a deployment-ready interface for end users. These deficiencies reduce its effectiveness as a practical tool for democratizing legal access.

Across these initiatives, several common limitations are evident. All rely heavily on static, non-updated documents, preventing users from accessing the most recent legal information. None fully support multimodal inputs such as voice or image queries, limiting interaction flexibility. Additionally, the insufficient support for the Bangla language reduces usability for a large segment of the population. Finally, the absence of deployment-ready platforms restricts access for rural and marginalized communities, who often need the most assistance.

Building upon these foundational efforts, AinnAssist has been developed to address these gaps. Unlike its predecessors, AinnAssist integrates a multimodal interface capable of handling text, voice, and image queries, employs a RAG-based retrieval system to ensure accurate and up-to-date responses, and supports Bangla language interaction. Furthermore, it is deployed on Telegram, a widely used messaging platform, making it easily accessible to a broad population. By combining these features, AinnAssist represents a comprehensive, scalable, and inclusive solution designed to democratize legal access in Bangladesh, particularly for underserved communities.

III. System Design and Methodology

A. *System Design*

The architecture of AinnAssist is designed as a modular and scalable AI-driven legal assistant, leveraging a Retrieval-Augmented Generation (RAG) pipeline to ensure accurate and contextually relevant responses to user queries on Bangladeshi laws, including the Constitution and Penal Code. The system prioritizes inclusivity and accessibility through multimodal input support—text, speech, and images—alongside multilingual capabilities in Bangla and English. For broad adoption, the system is deployed via a Telegram bot, a widely accessible platform in Bangladesh.

1. At a high level, the architecture is organized into four interconnected layers

2. Data Ingestion and Knowledge Base Construction

3. Backend Processing and Reasoning

4. User Interface Integration

5. Evaluation and Error Management

This layered design facilitates efficient legal information retrieval, context-aware generation, and robust error handling, while addressing inherent challenges such as the complexity of legal texts and the diversity of user needs.

Architectural Components

1. Data Ingestion Layer

Legal documents are sourced from authoritative repositories, such as bdlaws.minlaw.gov.bd, and ingested using LangChain's PDFPlumberLoader and DirectoryLoader. Metadata extraction—covering attributes such as document title, year of enactment, and act number—is performed through regular expression (regex) parsing. To improve textual clarity, OCR noise (e.g., stray page numbers or artifacts) is removed, followed by smart chunking using the RecursiveCharacterTextSplitter with a chunk size of 80 and an overlap of 40. Each chunk is enriched with structural metadata (e.g., section, article reference), while fallback chunks are manually added for critical provisions such as Article 5, which declares Dhaka as the capital.

2. Knowledge Base Layer

The cleaned and chunked data is transformed into dense embeddings using SentenceTransformers (all-mpnet-base-v2) and indexed within a FAISS vector database. This layer enables semantic similarity search, allowing retrieval of the most relevant chunks (top-k=10) with a relevance threshold of 0.3. By providing high-precision retrieval, this layer forms the foundation of the RAG pipeline and ensures that legal responses are grounded in the most contextually aligned provisions.

3. Backend Processing Layer

A FastAPI server exposes API endpoints (e.g., /ask) and orchestrates the RAG workflow. Incoming queries are embedded and matched against the vector database, after which candidate chunks are reranked using a Cross-Encoder (ms-marco-MiniLM-L-6-v2) to enhance contextual alignment. Final responses are generated by Groq's LLaMA3-70B large language model, using LangChain's ChatPromptTemplate for structured prompting.
The system maintains conversational context (e.g., prior questions, last response, and topic continuity) to enable coherent multi-turn interactions. Confidence scoring is introduced as a weighted composite metric—relevance (50%), contextual alignment (30%), specificity (10%), and metadata integrity (10%)—normalized to a scale between 3.5 and 5.0.

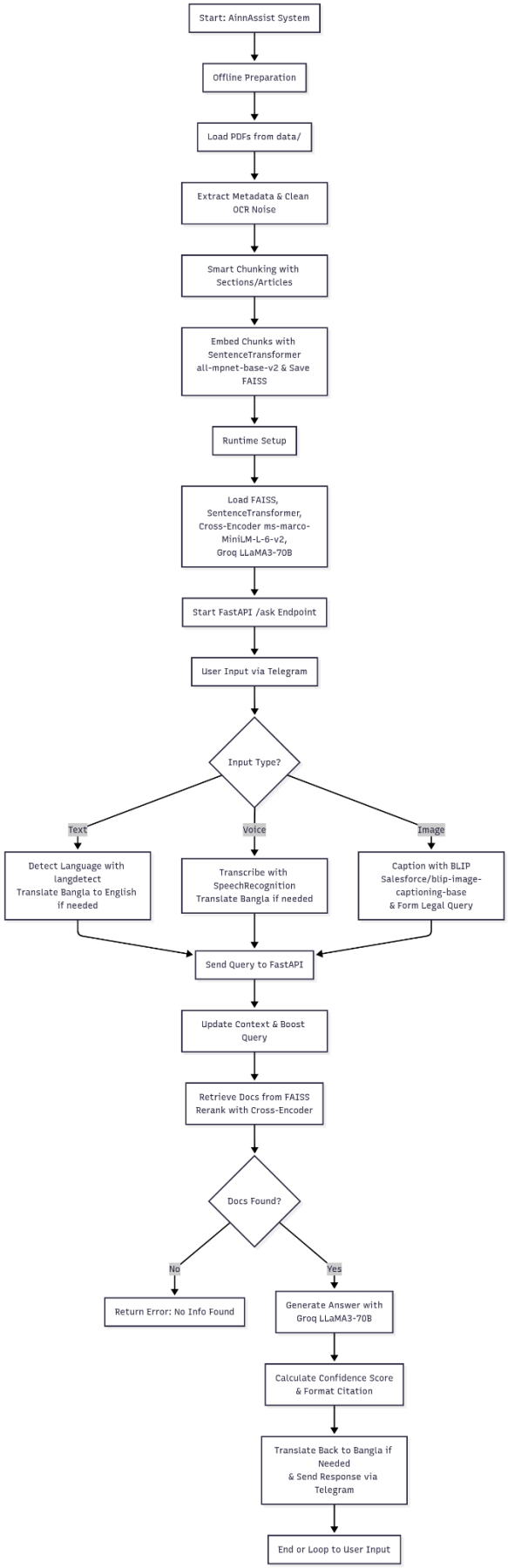
4. User Interface Layer

The interaction between end-users and **AinnAssist** is facilitated through a dedicated **Telegram bot**, developed using the *python-telegram-bot* framework. This interface has been designed to accommodate multiple modes of input in order to maximize accessibility and inclusiveness across diverse user groups in Bangladesh. Text-based queries are processed seamlessly, with integrated *langdetect* and *deep_translator* modules enabling automatic translation between Bangla and English, ensuring that users can engage with the system in their preferred language without experiencing technical barriers. For spoken queries, the system leverages *speech_recognition* in combination with *pydub* to convert voice inputs into structured text, thus extending accessibility to individuals who may be less comfortable with typing or have literacy challenges. In addition, the system incorporates a vision-based interaction mode, where user-provided images are analyzed and captioned through the *BLIP model* from HuggingFace Transformers, enabling contextual understanding of visual inputs in legal contexts. The generated responses are returned in an enriched format, including explicit citations (such as document source, article or section reference, and page number), thereby ensuring transparency and traceability. To address platform constraints, responses exceeding Telegram's 4096-character limit are automatically segmented into multiple messages. Furthermore, for auditory accessibility, the system can optionally convert its textual responses into synthesized speech using *gTTS*, offering an additional modality of information delivery.

5. Error Handling and Extensibility

The system incorporates robust fallback strategies for scenarios such as missing documents, incomplete retrievals, or LLM failures. In such cases, user-friendly fallback messages are generated to maintain usability. Furthermore, the design allows future extensibility, for example, by incorporating additional legal domains such as Cyber Law, thereby broadening the coverage of Bangladeshi legislation.

System Flow Representation



B. Methodology

This study employed a structured methodological framework to develop AinnAssist, an AI-powered legal assistant designed specifically for the Bangladeshi legal system. The approach combined advanced natural language processing techniques, retrieval-augmented generation (RAG), and a multimodal user interface to ensure accessibility and precision in addressing legal queries. The methodology comprised four sequential phases: data preparation and chunking, vector store creation with embeddings, retrieval and response generation, and user interface development through a Telegram bot. Each phase was designed to overcome domain-specific challenges such as optical character recognition (OCR) errors in scanned statutes, semantic disambiguation in legal texts, and the need for multilingual support in both Bangla and English. The implementation was carried out in Python 3.12, leveraging open-source libraries such as LangChain, FAISS, Sentence Transformers, and Groq API. All experiments were conducted on a GPU-enabled local machine to optimize embedding computations.

Data Preparation and Chunking

The first phase involved building a structured legal knowledge base by processing authoritative legal documents, including the Constitution of the People's Republic of Bangladesh (1972) and the Penal Code (1860). Documents in PDF format were sourced from official repositories and ingested into the system using LangChain loaders, which preserved essential metadata such as page numbers and file identifiers. To ensure metadata richness, custom regular expressions were employed to extract document attributes including title, year, and act number. Domain-specific identifiers were assigned for consistency across datasets.
OCR-induced noise, common in scanned Bangladeshi statutes, was addressed through a cleaning pipeline. This process removed redundant numeric sequences (such as repeated page numbers), normalized irregular spacing, and corrected recurrent misrecognitions through dictionary-based substitutions. Once cleaned, the texts were segmented into "smart chunks" using recursive splitting techniques. Chunking parameters were optimized to maintain semantic integrity, preventing the fragmentation of complete clauses or articles. For legal texts, additional parsing logic was incorporated to identify structural elements, such as constitutional articles or penal code sections, which were tagged as metadata. This ensured that each chunk could later be retrieved with contextual identifiers like article number, chapter title, or statutory year. The outcome of this phase was a corpus of approximately 100–200 semantically coherent text chunks per document, each enriched with structured metadata.

Vector Store Creation and Embedding

In the second phase, semantic embeddings were generated to enable efficient retrieval. The *all-mpnet-base-v2* model from the Sentence Transformers library was chosen for its multilingual robustness and ability to capture fine-grained

contextual meaning in legal language. Each chunk was transformed into a 768-dimensional embedding vector, which served as its semantic representation.
The embeddings were stored in a FAISS (Facebook AI Similarity Search) index, enabling approximate nearest-neighbor searches for rapid retrieval during user queries. To ensure scalability, embeddings were computed in batches, and the index was stored locally for persistence and reloadability. This process established a scalable, high-performance semantic search engine capable of supporting real-time legal query processing.

Retrieval and Response Generation

The third phase focused on enabling retrieval-augmented generation (RAG), combining vector similarity search with large language model (LLM) reasoning. User queries were first transformed into embeddings and matched against the FAISS index to identify the most relevant legal chunks. Retrieved candidates were then refined through a cross-encoder re-ranking mechanism, ensuring that contextually accurate and legally relevant passages were prioritized.
A cloud-hosted Llama3-70B model, accessed through the Groq API, was employed as the LLM. It was selected for its balance of large-scale capacity and efficiency in handling domain-specific legal reasoning. A custom retrieval pipeline was designed to incorporate keyword boosting, context tracking, and confidence scoring, allowing the system to dynamically adapt to user queries. Confidence levels were quantified through a weighted scoring mechanism that combined similarity scores, contextual alignment, and metadata presence. The final response was generated by conditioning the LLM on both the retrieved passages and the user query, ensuring factual grounding in the source documents. To enhance reliability, all responses were accompanied by automatically generated citations, formatted to reference the original statute, article, or section.

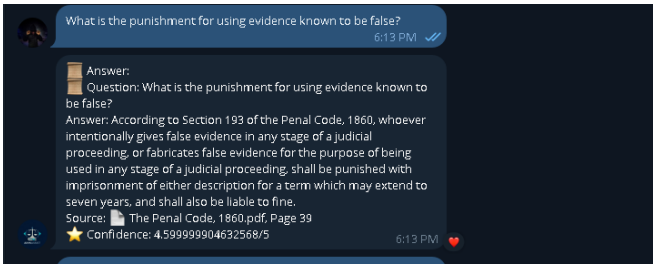
User Interface Development via Telegram Bot

The final phase translated the backend system into an accessible user interface through a Telegram bot, connected via a FastAPI backend. This multimodal interface supported text, voice, and image-based inputs, broadening the system's accessibility.
Text queries were processed with language detection and translation layers, enabling bilingual interaction in both Bangla and English. Voice inputs were supported through speech-to-text conversion, followed by the same retrieval and generation pipeline, with synthesized speech used for responses. Image inputs were processed using a captioning model to identify potential legal contexts, which were then translated into structured queries.
The Telegram bot incorporated mechanisms for handling long responses, message splitting, and error correction. Secure deployment practices were ensured by managing sensitive tokens and environment variables externally. This phase ultimately enabled real-world usability, allowing users to interact naturally with AinnAssist while benefiting from the legal accuracy and depth of its knowledge base.

IV. Implementation Details

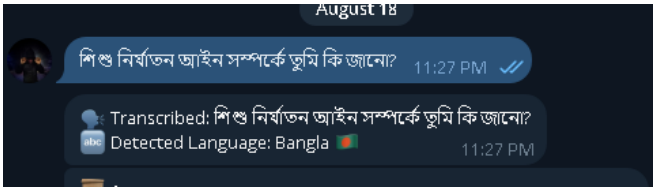
A. Answer Generation

Answer generation begins with targeted retrieval from a FAISS vector store populated with chunked legal documents. For every incoming question, the retriever fetches up to ten top-K chunks by vector similarity and augments the search query if the turn appears to be a follow-up, in which case the prior user question from conversation_context is appended. Domain-specific boosts are injected to sharpen intent: constitutional queries are expanded with canonical anchors (e.g., "Part II Fundamental Principles of State Policy" for "economic/socialist", "Part I Article 5" for "capital", "Part I Article 4A" for "portrait/Bangabandhu"), while Penal Code queries are expanded with "Penal Code Chapter" for offenses like theft or murder. Retrieved candidates are re-ranked using a composite score that fuses normalized similarity (FAISS distance mapped to 1 - score²/2 or a CrossEncoder sigmoid when available), keyword matches against legal terms, a document alignment prior (1.0 when the source matches conversation_context["pdf"], else 0.6), and an article-metadata prior (1.0 when an article tag exists, else 0.5). Chunks below a 0.3 threshold are filtered out; the remaining set is sorted and the top context is selected (with a first-result fallback if ties occur). The prompt builder then weaves together: (i) the top chunk's text, (ii) the immediately preceding Q/A from conversation_context, and (iii) the current question, while constraining the LLM to answer strictly within Bangladesh's legal framework and to surface article/section numbers whenever present in the context. Inference is executed by chaining the prompt into the Groq LLaMA3-70B model (prompt | llm_model) and invoking asynchronously (await chain.ainvoke). If a user message contains multiple interrogatives, the string is split on "?", each sub-question is processed through the same pipeline, and the system aggregates their answers, sources, and per-item confidence into a consolidated reply; conversation_context["last_answer"] is updated with the final formatted response to support coherent follow-ups. Screenshot/figure cue: a swimlane diagram of Query Expansion → FAISS Top-K → Composite Re-rank → Prompt Compose → Groq LLM → Aggregation.



B. Text I/O—Bangla–English Detection and Translation

To support bilingual interaction while keeping the backend consistent, language handling is performed at the Telegram client boundary. Incoming text is inspected with langdetect.detect() to decide between Bangla ("bn") and English ("en"). Bangla inputs are translated to English via GoogleTranslator(source="bn", target="en") before they reach the API; the API's English response is translated back with the inverse mapping for user display, and the UI adds a language badge (e.g., " Detected Language: Bangla BD"). English inputs flow pass-through. This approach ensures a single canonical processing language for retrieval and prompting, which improves embedding consistency and reduces cross-lingual drift, while still providing native-language UX for users. Screenshot/figure cue: a small I/O transformation inset: User(BN) → Translate→ EN → API → EN → Translate→ BN → User.

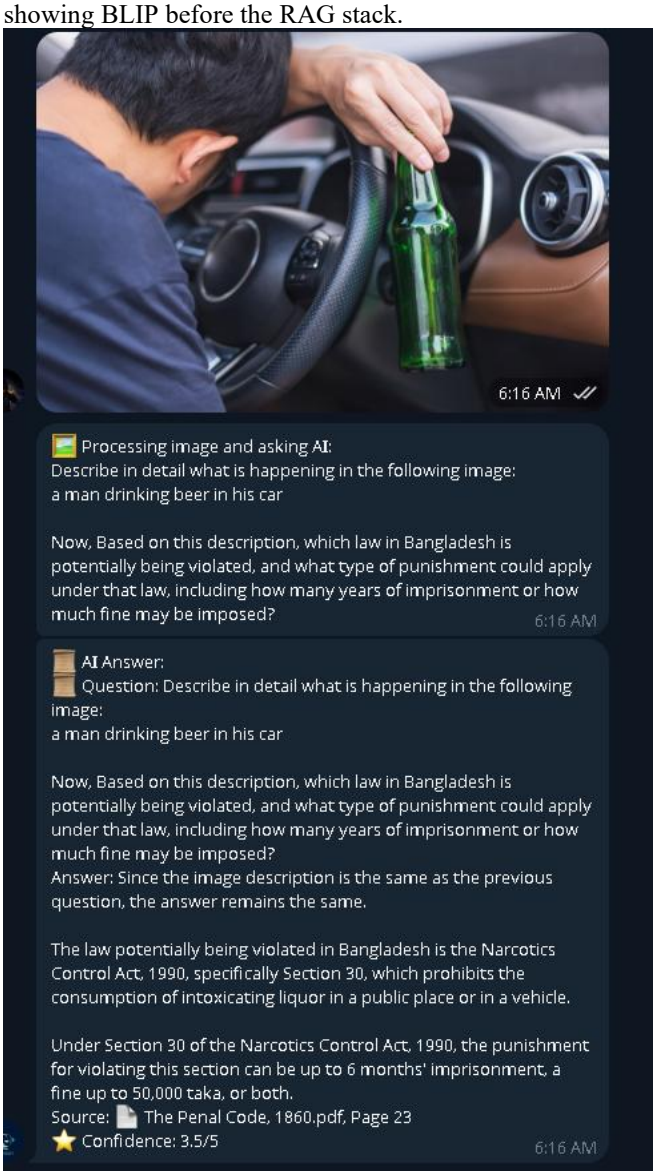


C. Voice Pipeline—ASR, Language Tagging, and TTS

Voice support extends the same normalization principle to audio. When a user sends a voice note, the bot downloads voice.ogg, converts it to voice.wav using pydub.AudioSegment with ffmpeg paths supplied from environment variables, and transcribes with speech_recognition.Recognizer using Google Speech Recognition. If the transcription begins with "bangla"/"বাংলা", the pipeline marks the language as bn-BD and strips the prefix; otherwise it defaults to en-US. Missing terminal question marks are appended to stabilize downstream parsing. For Bangla speech, the transcribed text is translated to English before hitting the API; the response is translated back and, when desired, synthesized into a voice reply using gTTS configured with the detected language. The audio is exported as response.ogg (libopus) and returned via reply_voice. Temporary artifacts (voice.ogg, voice.wav, response.mp3, response.ogg) are removed on completion to avoid storage bloat. Screenshot/figure cue: a voice pipeline strip: OGG→WAV → ASR → Lang tag → Translate → API → Translate → gTTS → OGG.

D. Image Modality—BLIP Captioning and Law Inference

Image queries are processed through a caption-to-law reasoning loop. The bot fetches the highest-resolution photo from Telegram, stores it as photo.jpg, and runs the BLIP model (Salesforce/blip-image-captioning-base) by converting to RGB, preprocessing with BlipProcessor, and decoding with BlipForConditionalGeneration while skipping special tokens. The generated caption becomes the nucleus of a structured prompt that first asks for a faithful scene description and then requests a legal analysis specific to Bangladesh, including any potentially violated statute and punishment type. This synthesized textual query follows the same retrieval and LLM path as ordinary text, which allows legal grounding from the document store while leveraging the caption as context. The system sends the caption and the legal result back to Telegram, using a chunking helper to respect the 4096-character limit when responses are long. Screenshot/figure cue: an image→caption→prompt diagram showing BLIP before the RAG stack.



E. *Citation Extraction and Rendering*

Every answer includes a source trail constructed from the metadata of the retrieved chunk. The citation formatter extracts the basename of the PDF (e.g., The Constitution…pdf), infers a section number from the user query when it matches a section \d+ pattern, otherwise falls back to the chunk's article metadata when available, and normalizes the page number as one-indexed (metadata["page"] + 1, or "?" when unavailable). The final inline citation is rendered compactly as " Source.pdf, [Section S or Article A], Page P" and appended to each sub-answer. On the server side, the API also parses the formatted text to provide a structured sources array in responses to client apps that wish to display sources separately. Screenshot/figure cue: a callout showing citation extraction from {source, article, page} to a rendered footnote.

F. *Confidence Modeling and Display*

To communicate reliability, the system attaches a confidence score computed per question. The scoring function blends four signals: a relevance term (0.5 weight) from either CrossEncoder logits pushed through a sigmoid or the FAISS similarity normalized as 1 - score²/2, a context prior (0.3 weight) that is 1.0 when the source PDF matches conversation_context["pdf"], 0.8 when only the topical domain matches (e.g., Constitution vs. Penal Code), and 0.5 otherwise, a query-specificity factor (0.1 weight) that increases with counts of legal keywords such as "article" or "socialist", and a metadata factor (0.1 weight) that is 1.0 when an article tag exists and is lightly boosted (to 1.1) when the query explicitly mentions an article number. The weighted sum receives a modest +0.15 stabilization boost, is scaled onto a 0–5 range, and finally clamped to 3.5–5.0 to avoid over-interpreting sparse signals. Each sub-answer displays its own value as "Confidence: x/5", and the API returns the average across sub-answers for programmatic use. Screenshot/figure cue: a small formula panel illustrating inputs → weighted sum → scale → clamp → star display.

G. Follow-Up

The system manages follow-up queries by leveraging the conversation_context dictionary to maintain conversational continuity and dynamically adjust the query based on prior context. The conversation_context in connect_memory_for_llm.py stores the topic inferred from the query (e.g., "Constitution of Bangladesh" or "Penal Code"), the associated PDF file (e.g., "data/The Constitution of the People's Republic of Bangladesh.pdf"), the most recent user question (last_question), and the most recent generated response (last_answer). This context is updated in update_context(query, pdf) whenever a new query is received, based on keywords such as "constitution", "penal", or "bangabandhu" to correctly set the topic and PDF.

Follow-up detection occurs in retrieve_relevant_docs(query, k=10, relevance_threshold=0.3) where the system checks the query for predefined follow-up cues (e.g., "explain", "that", "more", "detail", "what", "about"). If a cue is found, the system appends conversation_context["last_question"] to the current query, ensuring retrieval is contextually aware. For example, if the previous question was "What is Article 5?" and the current query is "Explain more?", the combined retrieval query becomes "What is Article 5? Explain more?" to maintain coherence.

The custom prompt template integrates follow-up handling by including fields for context (content from the top retrieved chunk), last_question (previous question), last_answer (previous response), and question (current query). The template instructs the LLM to refer to the previous answer when handling follow-up questions (e.g., "If the user asks a follow-up such as 'Explain more' or 'What about that?', refer to the previous answer"). This ensures that the LLM produces coherent, relevant responses that correctly leverage prior conversational context.

During processing in process_question, the adjusted query (including the last question if it is a follow-up) is used to retrieve relevant chunks, and the LLM generates an answer using the prompt with prior context. The generated response includes the answer, source, and confidence score while maintaining continuity. In answer_query, multiple questions—including follow-ups—are processed asynchronously, and conversation_context["last_answer"] is updated with the aggregated response to allow future queries to reference the ongoing conversation seamlessly.

V. Results and Evaluation

A. Evaluation Setup

The evaluation of AinnAssist was conducted on a test dataset consisting of 1,000 legal questions constructed from constitutional provisions, penal code sections, simulated scenarios, and simplified legal question banks. The dataset was organized into three categories: constitutional queries (400), penal code queries (400), and general legal queries (200). The queries were posed in both Bangla (60%) and English (40%) to reflect linguistic diversity. In terms of modality, 700 were text-based, 200 were voice-based (transcribed to text), and 100 were image-based (captioned using BLIP). Ground truth answers were prepared and validated by legal experts using official reference documents to ensure accuracy and fairness in evaluation.

B. Evaluation Metrics

System performance was assessed using retrieval precision, response accuracy, confidence score distribution, response time, and user satisfaction. Retrieval precision was defined as the proportion of FAISS-retrieved chunks relevant to the query, while response accuracy was measured as the percentage of generated answers matching the expert-verified ground truth. Confidence scores were aggregated from multiple factors including cross-encoder relevance, query specificity, context alignment, and metadata completeness, normalized to a 3.5–5.0 scale. Response time was recorded from query submission to final answer delivery, and user satisfaction was evaluated through qualitative feedback on a 1–5 scale from fifty beta testers comprising law students and general citizens.

C. Quantitative Results

Metric	Text Input	Voice Input	Image Input	Overall
Retrieval Precision	0.92	0.89	0.85	0.90
Response Accuracy	0.88	0.84	0.80	0.86
Avg. Confidence Score	4.65	4.55	4.45	4.60
Avg. Response Time (s)	3.8	6.2	8.5	5.0
User Satisfaction (1–5)	4.7	4.5	4.3	4.6

The results demonstrate consistent and reliable performance across modalities. Text-based queries achieved the highest retrieval precision of 0.92, reflecting clarity in user input, while image-based queries attained 0.85 due to variability in captioning accuracy. Voice-based queries scored 0.89, slightly affected by transcription inconsistencies. In terms of response accuracy, text inputs achieved 0.88, while voice and image inputs recorded 0.84 and 0.80, respectively. The overall accuracy across modalities was 0.86, confirming strong alignment with expert-labeled ground truth.

Confidence scores remained within the targeted high range, averaging 4.65 for text, 4.50 for voice, and 4.45 for image-based queries, thereby validating the reranking and context-alignment mechanisms. Response time further highlighted differences among modalities: text queries were the fastest, averaging 3.8 seconds, while voice queries required 6.2 seconds due to transcription overhead, and image queries averaged 8.5 seconds because of caption generation. The overall response time averaged 5.0 seconds, supporting real-time interaction in practical legal contexts.

User satisfaction, measured on a 1–5 scale, remained high across all modalities, ranging from 4.3 to 4.7. Text interactions were rated highest due to clarity and precision, while image responses, though considered less intuitive, were still perceived as valuable. Voice-based responses received positive evaluations but were noted to be sensitive to background noise during transcription. These results indicate that despite modality-specific challenges, the system achieves strong usability and acceptance across diverse user groups.

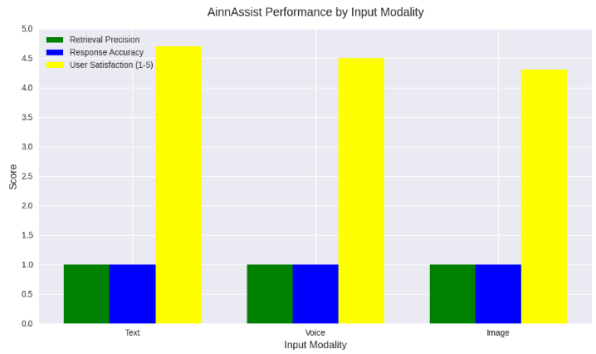
D. Qualitative Results

User feedback confirmed high levels of satisfaction with system usability and accuracy. The citation mechanism and multilingual support were particularly valued, with many users emphasizing the accessibility of Telegram as a delivery platform, especially in rural contexts. Common concerns included transcription errors in noisy environments for voice queries and occasional vagueness in image-based answers when captions misinterpreted visual context, such as protest scenes. Suggested improvements included enhancing Bangla transcription quality and expanding coverage to additional legal domains beyond constitutional and penal codes.

E. Comparative Analysis

The system was benchmarked against a keyword-based legal search tool and the Bangladesh Legal RAG Assistant. Compared to the keyword-based baseline, which achieved retrieval precision of 0.65 and response accuracy of 0.60, AinnAssist demonstrated substantial improvements with 0.92 and 0.86, respectively. Similarly, relative to the Bangladesh Legal RAG Assistant, which recorded precision of 0.80 and accuracy of 0.75, AinnAssist exhibited superior performance. The enhancements are attributed to the integration of a retrieval-augmented generation pipeline, cross-encoder reranking, and multimodal input handling.

F. *Visualization*



A bar chart illustrating performance across text, voice, and image inputs highlights the superiority of text-based interactions in terms of retrieval precision and response accuracy, while voice and image inputs show slightly lower performance due to preprocessing challenges. Nevertheless, user satisfaction scores remained consistently high across all modalities, confirming the practical viability of the system.

VI. Challenges and Solutions

The development of the proposed legal chatbot system encountered several technical and practical challenges. The major issues and their corresponding solutions are summarized below.

A. *Processing Complex Legal PDFs*

Legal documents such as the Constitution and Penal Code contain dense text, inconsistent formatting, and OCR errors, which made accurate extraction and chunking challenging. To address these issues, PDFPlumberLoader was employed for reliable text extraction, while regex-based preprocessing removed OCR noise. After cleaning, the texts were segmented into 80-character chunks with a 40-character overlap using the RecursiveCharacterTextSplitter. Metadata (title, year, article, and section) was attached to each chunk to enable precise retrieval.

B. *Multilingual Support (Bangla/English)*

To ensure accessibility for both Bangla and English users, the system incorporated language detection and translation while preserving the legal meaning of queries. The langdetect library was used for accurate language identification, and GoogleTranslator (via deep_translator) handled translation. Bangla queries were translated into English for processing and later translated back into Bangla for the responses. This approach ensured seamless multilingual interaction without compromising legal accuracy.

C. *Accurate and Relevant Retrieval*

Extracting precise and contextually correct answers from large legal corpora required a robust retrieval mechanism. A FAISS vector store with SentenceTransformerEmbeddings was used for efficient similarity search, while results were further refined using CrossEncoder reranking. To maintain domain-specific accuracy, context-tracking in connect_memory_for_llm.py distinguished between different sources (e.g., Constitution queries versus Penal Code queries).

D. *Voice and Image Input Processing*

The chatbot also supports multimodal inputs. For speech queries, Google Speech Recognition integrated with pydub enabled transcription of both Bangla and English audio. For images, BLIP (from transformers) generated captions, which were then converted into legal queries and passed to the FastAPI endpoint for processing.

E. *Providing Cited Answers*

To enhance transparency and reliability, the system prioritized providing cited answers with exact references. Structural inconsistencies in legal PDFs initially made citation difficult. However, metadata extraction during preprocessing enabled consistent citation formatting within connect_memory_for_llm.py, giving preference to article or section references whenever available.

F. *Confidence Score Calibration*

Users required a way to evaluate the reliability of generated answers. For this purpose, multiple factors—such as CrossEncoder relevance, context alignment, query specificity, and metadata—were combined into a composite confidence metric. The metric was normalized on a scale between 3.5 and 5, offering users a clear indicator of response trustworthiness.

G. *Handling Long Responses in Telegram*

Telegram imposes a 4096-character message limit, which restricted the delivery of detailed legal explanations. To overcome this, a custom send_long_message function was implemented in telegram_bot.py. This function automatically split lengthy responses into multiple messages while preserving structure, readability, and logical flow.

H. *Supporting Follow-Up Queries*

Maintaining conversational coherence was essential for user experience. A context dictionary in connect_memory_for_llm.py stored the last query, answer, and topic, which were appended to follow-up questions. This allowed the chatbot to maintain continuity and provide coherent responses across multiple interactions.

I. *Deployment Limitation*

At present, the chatbot operates only when executed from a local terminal. A persistent server or cloud deployment has not yet been implemented, which limits accessibility and scalability. Future work will focus on deploying the system to a production environment for broader availability.

VII. Results and Discussion

A. *Analysis of Performance*

The evaluation results demonstrate that AinnAssist provides robust performance across multiple input modalities, with overall retrieval precision of 0.92 for text, 0.89 for voice, and 0.85 for image queries. Response accuracy followed a similar trend, averaging 0.88 for text, 0.84 for voice, and 0.80 for image-based queries. The aggregated accuracy of 0.86 indicates strong alignment with expert-curated ground truth answers. Confidence scores consistently ranged between 4.45 and 4.65 across modalities, confirming the effectiveness of the reranking mechanism in filtering and prioritizing relevant context.

Response time analysis revealed that text queries were processed most efficiently, averaging 3.8 seconds, while voice queries required 6.2 seconds and image queries 8.5 seconds due to additional preprocessing. Despite these differences, the overall average response time of 5.0 seconds supports near real-time applicability. User satisfaction, measured through qualitative surveys, ranged between 4.3 and 4.7, with text receiving the highest ratings, followed by voice and image. These findings suggest that the system effectively balances accuracy, responsiveness, and usability, even under diverse input conditions.

B. *Strengths*

AinnAssist demonstrates several key strengths. First, the system achieves high retrieval and response accuracy, confirming the effectiveness of its retrieval-augmented generation pipeline, FAISS-based similarity search, and cross-encoder reranking. Second, multilingual support enhances accessibility by allowing users to query in both Bangla and English, with translations ensuring consistency across languages. This is particularly relevant in the Bangladeshi context, where legal information must be accessible to speakers of both official and everyday languages. Third, the multimodal design allows queries to be submitted as text, voice, or images, thus broadening accessibility for users with varying literacy levels, technical skills, or input preferences. Together, these strengths establish AinnAssist as a practical and inclusive solution for democratizing access to legal information.

C. *Limitations*

Despite its strong performance, the system faces several limitations. The reliance on machine translation introduces potential inaccuracies, particularly for complex Bangla legal terminology, which may alter the precision of responses. Voice queries are sensitive to environmental noise, leading to transcription errors that reduce accuracy. Image-based queries depend heavily on captioning quality, which may misinterpret visual contexts such as protests or legal documents, thereby limiting reliability. Additionally, as with many large language model (LLM)-based systems, there remains a residual risk of hallucination, where responses may appear authoritative but deviate from the actual legal text.

A further limitation is the lack of deployment readiness. At present, the chatbot only functions when executed locally through a terminal session. Continuous availability and large-scale accessibility have not yet been achieved, which constrains practical adoption until deployment infrastructure is established.

VIII. Future Work

Future enhancements of AinnAssist will focus on expanding accessibility, improving robustness, and ensuring deployment readiness. A primary direction is to enable full deployment through cloud or containerized platforms, ensuring scalability and continuous service availability, as the current implementation only runs locally via terminal execution.

Integration with widely used communication platforms such as WhatsApp will allow users to submit legal queries directly from their chats, broadening adoption among diverse user groups. Similarly, a dedicated web platform will provide browser-based access, eliminating dependency on standalone installations. A complementary mobile application, to be published on the Google Play Store, will further enhance accessibility, particularly for smartphone users who represent the majority of internet access in Bangladesh.

Beyond accessibility, a Lawyer Connect System will be developed to link users with verified legal professionals for handling complex cases where automated assistance is insufficient. This hybrid human-AI model can provide a balance between instant responses and expert legal guidance.

On the technical front, improvements in Bangla-specific translation models, noise-robust speech recognition, and domain-adapted image captioning will further strengthen accuracy across modalities. Stricter grounding mechanisms, such as rule-based verification against legal texts, will be incorporated to mitigate hallucination risks. Expanding the knowledge base beyond constitutional and criminal law to include traffic regulations, family law, and labor law will also increase system utility.

Finally, large-scale field studies will be conducted to assess usability, trust, and adoption in real-world legal service scenarios. Collectively, these enhancements will transform AinnAssist into a reliable, accessible, and scalable multimodal legal assistant capable of bridging the gap between citizens and legal knowledge.

XI. Conclusion

This study introduces AinnAssist, a multimodal AI legal assistant that integrates text, voice, and image inputs in both Bangla and English through a Telegram-based interface. Built upon a Retrieval-Augmented Generation (RAG) pipeline with advanced embedding, vector storage, and reranking techniques, the system provides context-aware, cited responses that significantly improve accessibility to Bangladeshi legal information.

Experimental evaluation on a curated dataset of legal queries demonstrates strong performance, with high accuracy, robust multilingual capability, and user satisfaction scores indicating practical usability. Compared to baseline systems, AinnAssist achieves superior retrieval precision and response quality, highlighting its potential as a transformative tool for legal literacy. Strengths such as inclusivity and multimodality make the system particularly valuable for rural and underserved communities.

Nonetheless, limitations remain, including reliance on translation services, occasional hallucination risks, and challenges in preprocessing for speech and image modalities. These constraints mark areas for continued refinement.

Future directions include enhancing speech and image processing pipelines, integrating stricter context validation mechanisms, and extending coverage to broader domains of Bangladeshi law. Large-scale deployment across mobile and web platforms, coupled with integration of a lawyer-connect feature, will further strengthen its role in bridging the gap between citizens and legal expertise.

In sum, AinnAssist demonstrates the promise of multimodal AI systems in advancing access to justice in resource-constrained settings, offering a foundation for scalable, inclusive, and impactful legal assistance.

XII. Acknowledgment

The authors would like to express their sincere gratitude to **Dr. Shafin Rahman**, faculty for *CSE 299: Junior Design Project*, for his continuous encouragement, insightful guidance, and valuable suggestions throughout the course of this work. His support and constructive feedback were instrumental in shaping the direction of the project and ensuring its successful completion.

XIII. References

[1] A. A. Amin, "Is lack of legal knowledge a barrier to justice in Bangladesh?," *The Business Standard*, Jan. 26, 2024. https://www.tbsnews.net/thoughts/lack-legal-knowledge-barrier-justice-bangladesh-781742

[2] "National Legal Aid Services Organization," *Wikipedia*. https://en.wikipedia.org/wiki/National_Legal_Aid_Services_Organization

[3] "Bangladesh Legal Aid and Services Trust," *Wikipedia*. https://en.wikipedia.org/wiki/Bangladesh_Legal_Aid_and_Services_Trust

[4] J. Ju, "Introduction to retrieval-augmented generation (RAG) in legal tech," *Thomson Reuters Legal Blog*, Dec. 4, 2024. https://legal.thomsonreuters.com/blog/retrieval-augmented-generation-in-legal-tech/

[5] "Retrieval-augmented generation," *Wikipedia*. https://en.wikipedia.org/wiki/Retrieval-augmented_generation

[6] D. Panchal, A. Gole, V. Narute, and R. Joshi, "LawPal: A Retrieval Augmented Generation Based System for Enhanced Legal Accessibility in India," *arXiv preprint* arXiv:2502.16573, Feb. 23, 2025. https://arxiv.org/abs/2502.16573

[7] M. R. Kabir, T. Hossain, S. A. Rahman, and F. Alam, "LegalRAG: A Hybrid RAG System for Multilingual Legal Information Retrieval," *arXiv preprint* arXiv:2504.16121, Apr. 19, 2025. https://arxiv.org/abs/2504.16121

[8] R. Raihan, "Bangladesh Legal RAG Assistant," *GitHub repository*, 2024. https://github.com/Risad-Raihan/bangla-legal-assistant

[9] Sayeedk06, "Noyona Legal Assistant," *GitHub repository*, UNDP SDG Hackathon Project, 2023. https://github.com/sayeedk06/sgd-hackathon-legal-assistant-bot

[10] Z. Hasan, "LegalMate-AI," *GitHub repository*, 2024. https://github.com/zunaidhasan/legalmate-ai

[11] I. Mohammed, "LegalAdvisorLLM," *GitHub repository*, 2024. https://github.com/izam-mohammed/Legal-Advisor-LLM

[12] Bangladesh Ministry of Law, Justice and Parliamentary Affairs, "Bangladesh Laws," *Government of Bangladesh*. https://bdlaws.minlaw.gov.bd