UNIVERSITY SCHOOL OF AUTOMATION AND ROBOTICS
GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY
EAST DELHI CAMPUS, SURAJMAL VIHAR, DELHI- 110032
http://ipu.ac.in/eastcampusmain.php

Minor Project Report on

“ProdigalAI”
- KnowledgeAI Chatbot

Submitted in partial fulfilment of the requirements for the completion of minor project during 7th semester

Name: Kushagra Wadhwa
Enrollment Number: 08719011621

Under the supervision of

DECLARATION

I hereby declare that the Minor Project Report entitled "***KnowledgeAI Chatbot @ProdigalAI***" is an authentic record of work completed as a requirement of Minor Project Report during the period from "27/07/2024" to "31/07/2024" at **ProdigalAI** under the supervision of Nishchal Gaba, CEO of ProdigalAI

Date:11/11/2024

KUSHAGRA WADHWA
08719011621

Under the supervision of

Date:

PRODIGAL AI TECHNOLOGIES PVT LTD

Dear Kushagra Wadhwa,

We are glad to offer you an internship as an Industry Collaboration Minor Project Intern with Prodigal AI for a period of 3.5 months starting from 2nd September 2024 with a stipend of INR 0/Month.

Before beginning the internship, you will be required to sign confidentiality and non-disclosure agreements in such form and manner as acceptable to Prodigal AI. Any work of internship reports to be submitted to your institution on completion of internship shall be submitted only after a prior review by Prodigal AI and disclosure of any confidential and/or proprietary information in such reports shall be subject to the prior written consent of Prodigal AI.

During the course of the internship, Prodigal AI may require you to maintain such records as may be necessary to evidence the progress of your internship.

Please note that during the period of internship you will be required to abide by Prodigal AI's code of ethics and other rules and policies. Any violation of the Code of Ethics or rules or policies or if any of your actions are deemed detrimental to the interests of Prodigal AI will entitle Prodigal AI to terminate the internship with immediate effect. Please note that this internship is not an offer of employment or a precursor to an offer of employment at Prodigal AI.

Warm Regards,

Nishchal Gaba,
CEO,
PRODIGAL AI PTY LTD

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to everyone who contributed to the successful development of the Constitution Chatbot project.

Firstly, I would like to thank Nishchal Gaba, whose guidance and support have been invaluable throughout this project. Their expertise and feedback have greatly enhanced my understanding and shaped the direction of my work.

I am also grateful to my colleagues and peers for their encouragement and insightful discussions, which enriched my perspective and helped overcome various challenges along the way.

Special thanks go to the developers and communities behind the open-source tools and libraries used in this project, including OpenAI's Whisper for speech recognition, Streamlit for the interactive interface, and the developers of gTTS and Gemini Embeddings. Their resources have provided the foundation for implementing advanced functionalities and brought this chatbot to life.

Finally, I would like to acknowledge the unwavering support of my family and friends, whose encouragement motivated me throughout this project. Their belief in me has been instrumental in achieving this milestone.

Thank you to everyone who helped make this project a success

INDEX

1. Abstract

The Constitution Chatbot is a robust, interactive tool designed to enable users to query the bot and ask it questions about the Indian Constitution. By leveraging advanced natural language processing and machine learning algorithms, the chatbot provides an intuitive interface for retrieving information from dense and complex documents. Key technologies integrated into the chatbot include Whisper for transcription, Gemini for embeddings, and gTTS for text-to-speech responses, creating a seamless experience for voice and text interactions with document content.

2. Introduction

With the vast and growing volume of digital information, finding specific content within extensive documents like the Indian Constitution can be a daunting task. The complexity of legal language, combined with sheer document length, means that users often struggle to locate precise answers quickly.

The Constitution Chatbot addresses these challenges by providing an interactive, AI-powered tool designed to understand user queries in natural language and deliver accurate answers directly from the document. Users can type or speak their questions, enabling quick access to targeted information within the Indian Constitution without the need for manual searches. By understanding the intent and context behind questions, the chatbot provides a more intuitive and efficient way to access complex information, streamlining legal reference, research, and education.

This project leverages Streamlit to create a user-friendly interface, allowing users to interact with the chatbot seamlessly. Behind the scenes, it integrates various AI-driven models, including OpenAI's Whisper for speech-to-text conversion, natural language processing for contextual document indexing, and text-to-speech (TTS) capabilities for spoken responses. This combination of technologies transforms the Constitution Chatbot into a sophisticated virtual assistant, enabling users to interact with complex legal information in an accessible, conversational manner.

3. Literature Survey

The Constitution Chatbot is an advanced AI system integrating cutting-edge technologies in speech recognition, text-to-speech synthesis, embedding-based retrieval, and hybrid retrieval frameworks. Each component leverages state-of-the-art developments, enabling the chatbot to provide seamless voice-based interaction and precise information retrieval. Below, we delve deeper into these foundational technologies and their significance.

1. Speech Recognition

Technology: Whisper by OpenAI

Whisper is an advanced automatic speech recognition (ASR) model that has set new standards for transcription accuracy and versatility. Trained on an expansive and diverse dataset encompassing multiple languages, accents, and audio qualities, Whisper demonstrates robust capabilities in:

- **Multilingual transcription**: It supports transcription in a variety of languages, making it suitable for multilingual applications.
- **Noise resilience**: Whisper handles audio with significant background noise, a common challenge in real-world applications.
- **Low-resource adaptability**: Its robust training makes it effective even with lower-quality recordings or uncommon languages.

In the Constitution Chatbot, Whisper is the backbone for converting user speech into text. Its high accuracy ensures that user inputs are transcribed correctly, minimizing errors that could lead to irrelevant or incorrect responses. This reliability is crucial for systems interacting with legal or constitutional content, where precision is paramount.

Literature:

- Radford et al. (2022) highlighted Whisper's fine-tuning potential for domain-specific tasks, making it ideal for niche applications like constitutional queries.
- Studies on ASR in legal contexts emphasize the need for transcription systems that minimize semantic drift, a strength demonstrated by Whisper.

---

2. Text-to-Speech (TTS)

Technology: Google Text-to-Speech (gTTS)

gTTS is a Python library providing a simple yet powerful interface for generating natural-sounding speech from text. Key features include:

- **Language support**: It supports a wide range of languages and dialects.
- **Customizability**: Users can adjust speech speed and tone to suit application needs.
- **Accessibility**: As an open-source tool, it allows easy integration into larger systems.

In the Constitution Chatbot, gTTS enhances accessibility by providing voice responses. This feature caters to users with disabilities, such as visual impairments, or those who prefer auditory interactions. Additionally, voice output makes the chatbot more engaging and suitable for conversational use cases.

Literature:

- Shen et al. (2018) explored advancements in TTS systems, noting their importance in creating user-friendly interfaces. gTTS aligns well with these trends by combining simplicity and effectiveness.
- Applications of TTS in interactive systems show improved user satisfaction when voice-based features complement textual outputs.

3. Embedding Models

Technology: Gemini Embeddings

Embedding models convert text into numerical vectors that capture semantic meaning. Gemini Embeddings, specifically, excel in creating high-dimensional vector spaces optimized for similarity-based search. Their characteristics include:

- **Semantic representation**: Gemini Embeddings capture context beyond keywords, enabling the system to understand nuanced user queries.
- **Scalability**: The embeddings are efficient for large-scale datasets, making them suitable for indexing vast constitutional texts.
- **Domain adaptability**: Fine-tuning allows embeddings to align closely with specific domains, such as legal or constitutional con tent.

In the Constitution Chatbot, these embeddings are used to vectorize both the document corpus and user queries. By comparing these vectors, the chatbot retrieves the most relevant sections of the Constitution, even when queries are phrased conversationally or lack precise legal terminology.

Literature:

- Mikolov et al. (2013) introduced the foundational concepts of embeddings, which Gemini builds upon with enhanced contextual representation.
- Recent developments in legal AI emphasize embeddings for improved retrieval in complex domains, as noted by Zhong et al. (2021).

---

4. Hybrid Retrieval Systems

Technology: Retrieval-Augmented Generation (RAG)

Hybrid retrieval systems, such as RAG, combine vector-based semantic retrieval with traditional keyword-based methods. This dual approach ensures:

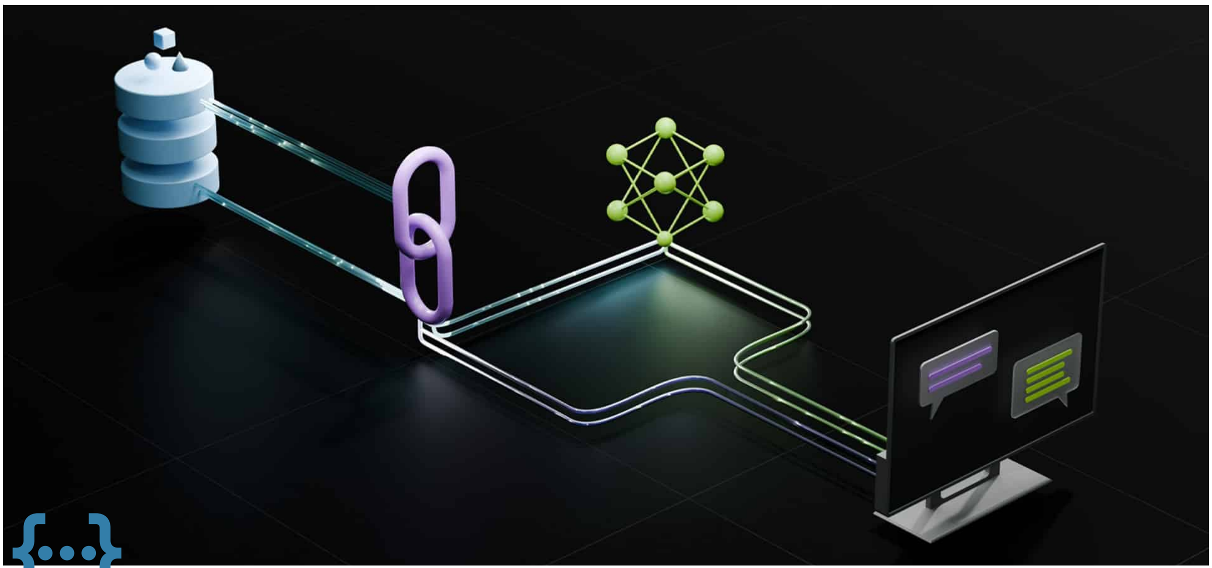
- **Precision**: Semantic search retrieves contextually relevant results.
- **Recall**: Keyword search captures exact matches, ensuring no crucial information is missed.
- **Dynamic query handling**: The system balances contextual understanding and direct matches, adapting to diverse query formulations.

The Constitution Chatbot integrates RAG to handle user queries effectively. When a user asks a question, the chatbot first searches its indexed corpus using embeddings to find relevant sections semantically. Simultaneously, keyword retrieval is used to cross-check for direct matches, providing a robust information retrieval mechanism.

Literature:

- Lewis et al. (2020) outlined the advantages of RAG in balancing retrieval efficiency and accuracy, highlighting its superiority over single-method approaches.
- Applications of RAG in legal contexts, such as summarizing case laws, show its effectiveness in handling large and complex document sets.

Articles on following topics were researched and written in accordance with the research conducted during the due course of the internship;



RAG Revolution🚨: Supercharging AI🤖 with Retrieval-Augmented Generation🧬

What is RAG?

RAG stands for Retrieval-Augmented Generation. It's a powerful technique in natural language processing that combines the strengths of traditional information retrieval systems (such as databases) with the capabilities of generative large language models (LLMs). It is the process of optimizing the output of a large language model. Large Language Models (LLMs) are trained on vast volumes of data and use billions of parameters to generate original output for tasks like answering questions, translating languages, and completing sentences. Retrieval Augmented Generation (RAG) is an architecture that augments the capabilities of a Large Language Model (LLM) like ChatGPT by adding an information retrieval system that provides grounding data . Adding an information retrieval system gives you control over grounding data used by an LLM when it formulates a response. RAG extends the already powerful capabilities of LLMs to specific domains or an organization's internal knowledge base, all without the need to retrain the model. It is a cost-effective approach to improving LLM output so it remains relevant, accurate, and useful in various contexts.

For Example, let's say I want to build a chatbot only related to the internal HR Policy documents of the organization then the grounding data of that LLM will be the HR policy documents of that organization and is private to the internal employees of the organization and is of no use to the other people outside the organization, this is where RAG is needed. Retrieval-Augmented Generation (RAG) is essential for integrating external knowledge into Large Language Model (LLM) outputs Retrieval-augmented generation (RAG) is an AI framework for improving the quality of LLM-generated responses by grounding the model on external sources of knowledge to supplement the LLM's internal representation of information. RAG is currently the best-known tool for grounding LLMs on the latest, verifiable information, and lowering the costs of having to constantly retrain and update them. RAG depends on the ability to enrich prompts with relevant information contained in vectors, which are mathematical representations of data. Vector databases can efficiently index, store and retrieve information for things like recommendation engines and chatbots. The RAG model consists of two main stages:

- **Retrieval Stage:** Information relevant to the given question is retrieved through a search engine.
- **Generation Stage:** Answers are generated based on the retrieved information.

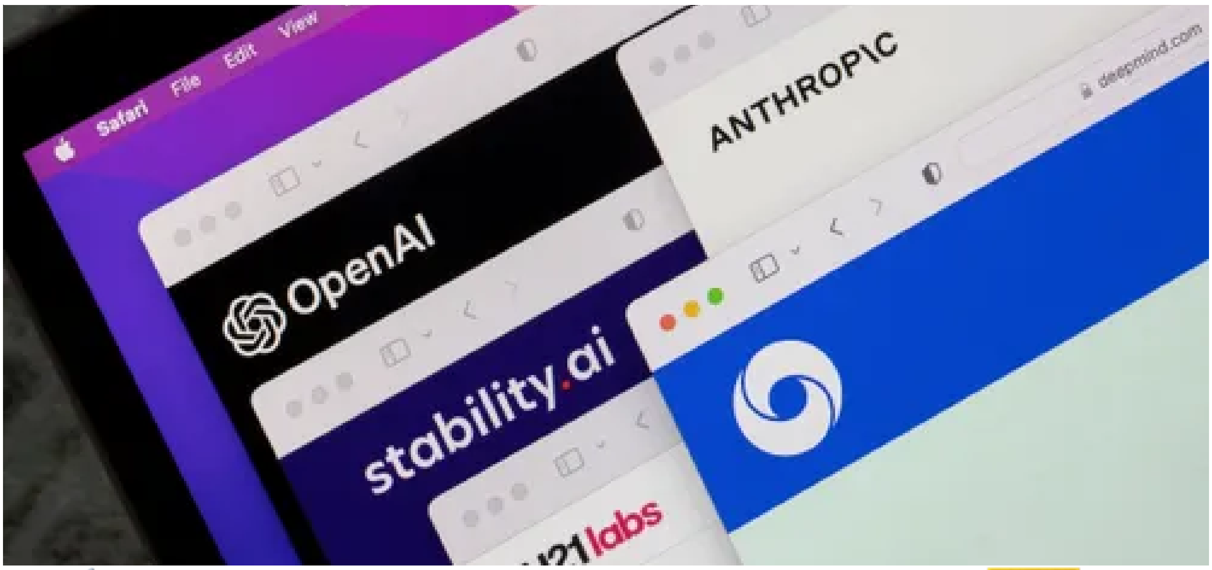
```
graph LR
    A[User Query] --> B[Retrieval System]
    B --> C[Knowledge Base]
    C --> D[Retrieved Information]
    D --> E[Language Model]
    E --> F[Generated Response]
    A --> E
```

💡 The above figure represent the working of RAG in the simplest way

How RAG Works?

RAG operates through a two-step process:

1. **Retrieval:** The system searches a knowledge base for relevant information. RAGs leverage powerful search algorithms to query external data, such as web pages, knowledge bases, and databases. Once retrieved, the relevant information undergoes pre-processing, including tokenization, stemming, and removal of stop words.
2. **Generation:** The retrieved information is used to augment the language model's knowledge, and is seamlessly incorporated into the pre-trained LLM; enabling it to generate more accurate and contextually relevant responses. This integration enhances the LLM's context, providing it with a more comprehensive understanding of the topic. This augmented context enables the LLM to generate more precise, informative, and engaging responses.



🪡Tailoring LLMs: The Art🖼️ of Fine-Tuning for Personal Data 🤫

Fine-tuning Large Language Models (LLMs) has revolutionized Natural Language Processing (NLP), offering unprecedented capabilities in tasks like language translation, sentiment analysis, and text generation. This transformative approach leverages pre-trained models like GPT-2, enhancing their performance on specific domains through the fine-tuning process.

In the ever-expanding realm of AI and machine learning, **Large Language Models (LLMs)** have taken center stage. From powering chatbots to enabling advanced research, LLMs like OpenAI's GPT or Meta's LLaMA are transforming industries. But with these powerful models comes a crucial question: **How do we make them more relevant, especially when dealing with personal or domain-specific data?** This is where **fine-tuning** enters the scene, evolving a generalist AI into a bespoke assistant.

Understanding How Pre-trained Language Models Work?

The Language Model is a type of machine learning algorithm designed to forecast the subsequent word in a sentence, drawing from its preceding segments. It is based on the Transformers architecture, which is deeply explained in our article about **How Transformers work.**

Pre-trained language models, such as GPT (Generative Pre-trained Transformer), are trained on vast amounts of text data. This enables LLMs to grasp the fundamental principles governing the use of words and their arrangement in the natural language.

The Craft of Fine-Tuning

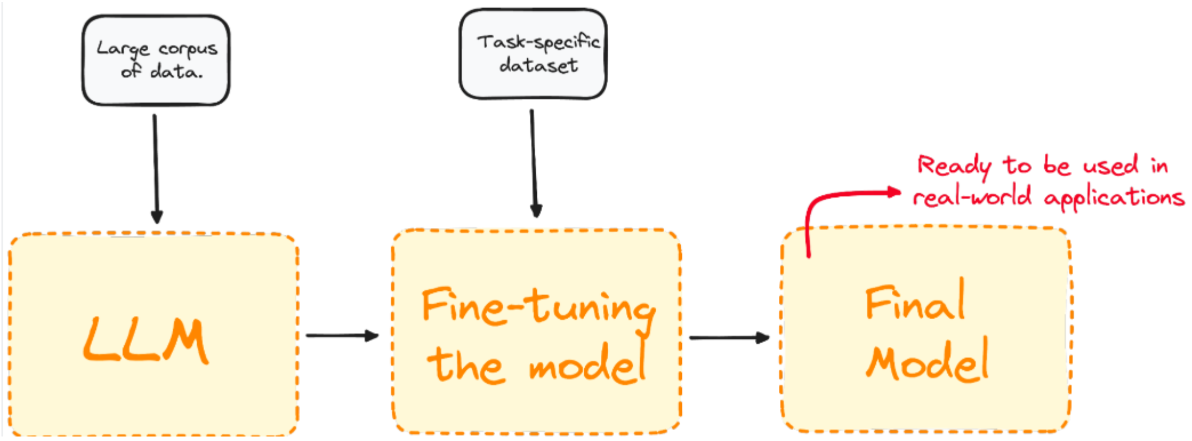
Think of LLMs as an intricate tapestry woven from billions of data points. At their core, they've been trained on a vast, diverse corpus—everything from literature to programming tutorials. But when you need an LLM to handle **specific tasks**—such as acting as an assistant to a legal expert, a healthcare provider, or even a niche marketing professional—**general knowledge isn't enough**. Fine-tuning is like tailoring that broad skillset, making the model more efficient and accurate for particular uses. Fine-tuning is the process of taking a pre-trained model and further training it on a domain-specific dataset.

But how exactly does one tailor an LLM for personal data?

What is fine-tuning?

Fine-tuning is a process in which a pre-trained model is re-trained on custom or specific data. A good example of a fine-tuned model is the *instruct* version of Mistral-7B, which is designed to be a conversational AI, i.e., a chatbot.

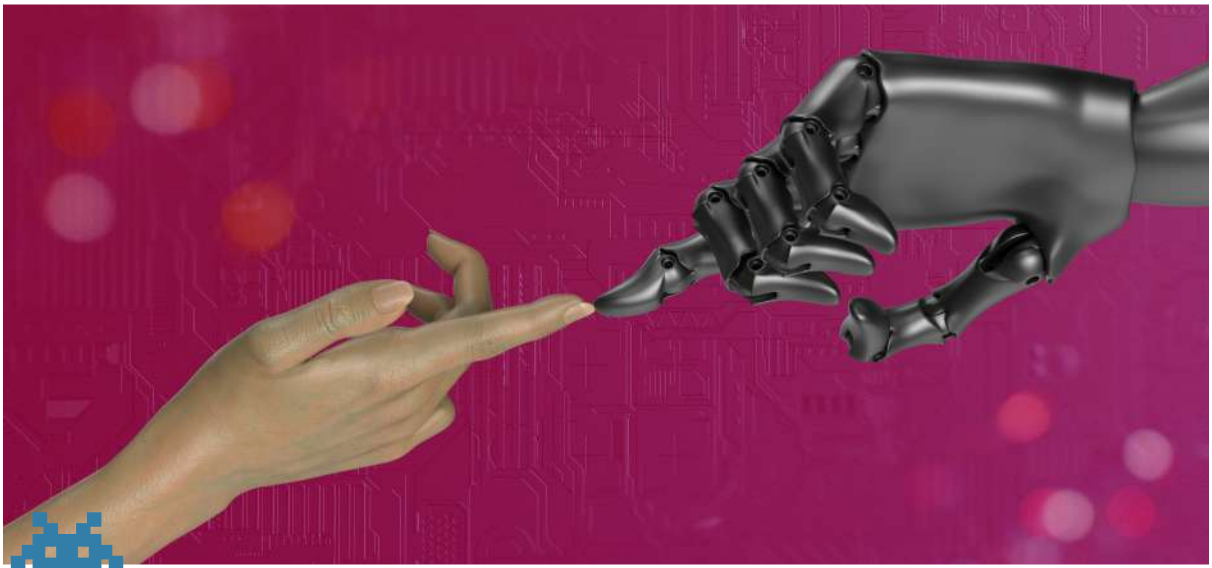
Fine-tuning can be done to simply change a model's responses using instruction datasets, such as with the *instruct* versions of Mistral, or to create a more industry-specific model. In this guide we will be creating a UbiOps-specific model which can answer questions about our product.



Fine-Tuning in Action 🎨{Types}

Fine-tuning can take several forms, depending on the complexity of the task. Here are three key methods:

1. **Few-Shot Learning**: This allows the model to understand a new task from a few examples. Suppose you want the LLM to generate marketing copy for products like personalized t-shirts. By providing a handful of product descriptions, the model can adapt and start producing more relevant outputs without extensive re-training.
2. **Supervised fine-tuning:** The most straightforward and common fine-tuning approach. The model is further trained on a labeled dataset specific to the target task to perform, such as text classification or named entity recognition.
3. **Transfer learning**: Even though all fine-tuning techniques are a form of transfer learning, this category is specifically aimed to allow a model to perform a task different from the task it was initially trained on. The main idea is to leverage the knowledge the model has gained from a large, general dataset and apply it to a more specific or related task.
4. **Domain-Specific Adaptation**: Imagine a researcher in astrophysics fine-tuning a model on **research papers** from arXiv or NASA's archives. The result? The LLM becomes a reliable assistant for generating hypotheses, summarizing research, or even writing scientific abstracts.
5. **Persona Creation**: Individuals can fine-tune models to mimic their communication style. Think about personal assistants or chatbots trained on an individual's tone, vocabulary, and preferences. Fine-tuned LLMs can



Reflection 70B🤖: Revolutionizing AI with Self-Correction and Superior Performance?

`Reflection-Llama-3.1-70B`

The **Reflection Llama-3.1 70B** is an open source LLM (Large Language Models) trained with a new technique **Reflection-Tuning** that teaches a LLM to detect mistakes in its reasoning and correct course.

The ideology behind this model "I've been thinking about this idea for months now. LLMs hallucinate, but they can't course-correct. What would happen if you taught an LLM how to recognize and fix its own mistakes?"

Hence the name, "Reflection" — a model that can reflect on its generated text and assess its accuracy before delivering it as outputs to the user.

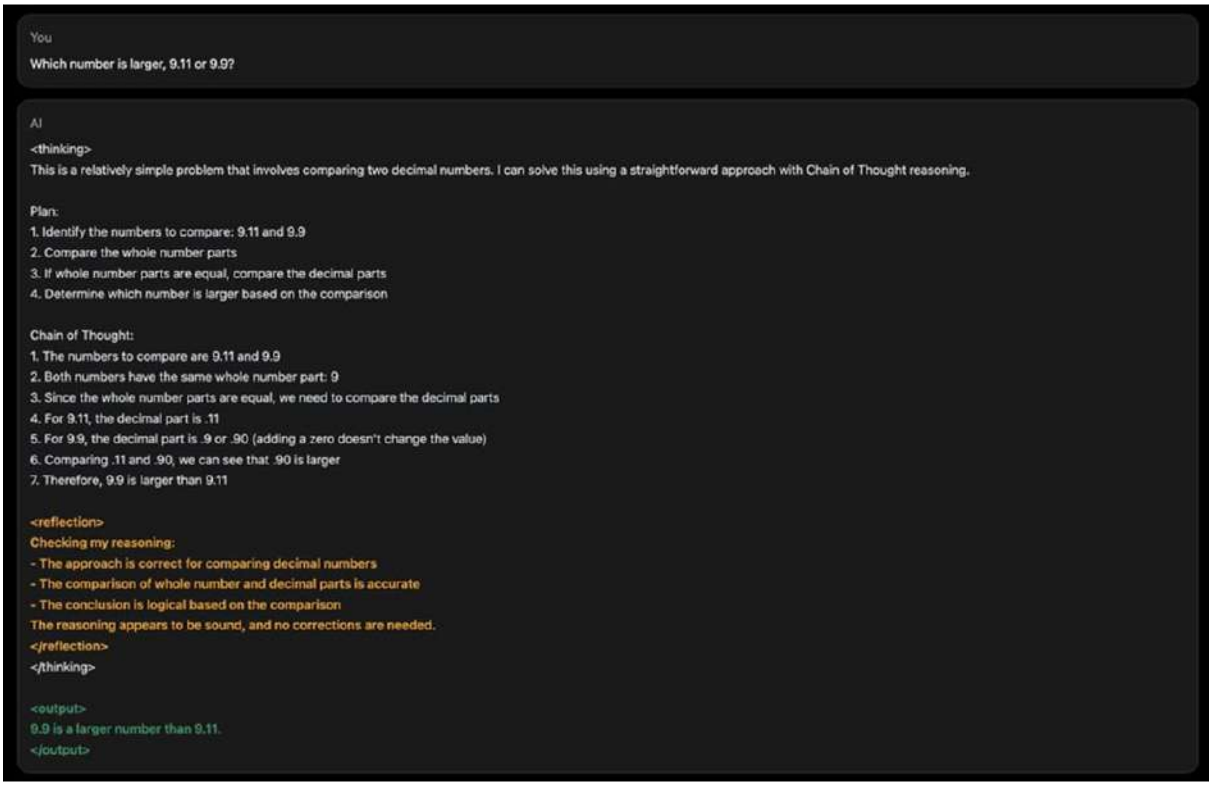
What is "Reflection 70b"*??*

Reflection 70B is based on Meta's open-source **Llama 3.1 70B** Instruct and utilizes the same code pipeline as other Llama models, but incorporates reflection tuning, a technique that allows the LLM to recognize mistakes and correct itself before finalizing the answer. Reflection 70B introduces new special tokens for inference and error correction, making it easier for users to interact with the model in a more structured way.

This feature enhances the model's ability to provide accurate and reliable responses by allowing it to identify and rectify potential errors during the inference process. By implementing this self-correction mechanism, Reflection 70B aims to reduce the occurrence of hallucinations and improve the overall quality of its outputs.

During inference, Reflection 70B outputs inferences in special tags, so that if an error is detected, it can be corrected in real time.

In addition, by separating the 'planning' phase from the answer generation stage, Reflection 70B appears to embrace the effectiveness of **the Chain-of-Thought** approach while making the output simpler and more concise for end users.



Training Methodology: How Does Reflection 70B Differ??🤔💭

The model was trained on synthetic data that is, a custom smaller dataset helping the model getting trained on a particular use case the synthetic datasets for fine-tuning models are faster, cheaper and consistently outperform general purpose models. Glaive's approach is to create synthetic datasets tailored to specific needs, allowing them to fine-tune models quickly and affordably.

This approach to model training offers several advantages. Firstly, it allows for more targeted and efficient training, as the model can focus on learning specific tasks or domains rather than attempting to cover a broad range of general knowledge. Secondly, the use of synthetic datasets can help overcome limitations in available real-world data, especially for specialized or niche applications. Lastly, this method can potentially lead to faster development cycles and reduced costs in AI model creation and deployment.

4. Problem Statement

Accessing specific information within large and dense documents, such as the Indian Constitution, is a significant challenge due to the document's complex language, length, and hierarchical structure. Traditional keyword search methods often lack the sophistication needed to interpret context or understand synonym, making it difficult to handle nuanced queries effectively. Simple keyword searches are prone to returning either overly broad or irrelevant results because they cannot grasp the deeper intent behind a query or locate synonyms accurately within a legal context.

The Constitution Chatbot addresses these limitations by introducing several key advancements:

1. **Conversational Interface for Natural Language Queries**: Rather than forcing users to remember specific terms or phrases, the chatbot allows them to phrase questions in a conversational way, just as they would ask a human expert. This natural language understanding enables users to ask complex or multi-faceted questions, and the bot interprets their intent without the need for exact keywords.
2. **Multi-Modal Interaction Through Text and Voice**: The chatbot offers flexibility by supporting both text and voice input, allowing users to engage through their preferred method. This multi-modal capability not only makes the chatbot more accessible to a wider audience but also improves usability for individuals with disabilities or those seeking hands-free interaction.
3. **Enhanced Retrieval Accuracy with Combined Indexing Techniques**: The chatbot improves the precision of search results by integrating both **vector embeddings** and **keyword-based indexing**. Vector embeddings allow the system to capture semantic relationships within the document, mapping similar concepts and phrases closely together even if they don't share exact terms. Combined with keyword-based indexing, this hybrid approach ensures that both direct matches and contextually similar content are retrieved, providing users with relevant answers even for nuanced or indirect queries.
4. **Contextual Understanding and Enhanced Query Interpretation**: By leveraging AI-driven models, the chatbot interprets the context surrounding each query, accounting for legal terms and cross-referencing related sections within the document. This contextual understanding ensures that answers are not only accurate but also relevant to the user's specific question, reducing the need for further refinement or additional searches.

Through these features, the Constitution Chatbot streamlines access to information in large legal texts, enabling users to quickly locate precise, contextually appropriate answers within a conversational framework.

5. Description of Training Modules

The Constitution Chatbot is powered by several key modules that work together to provide a seamless user experience:

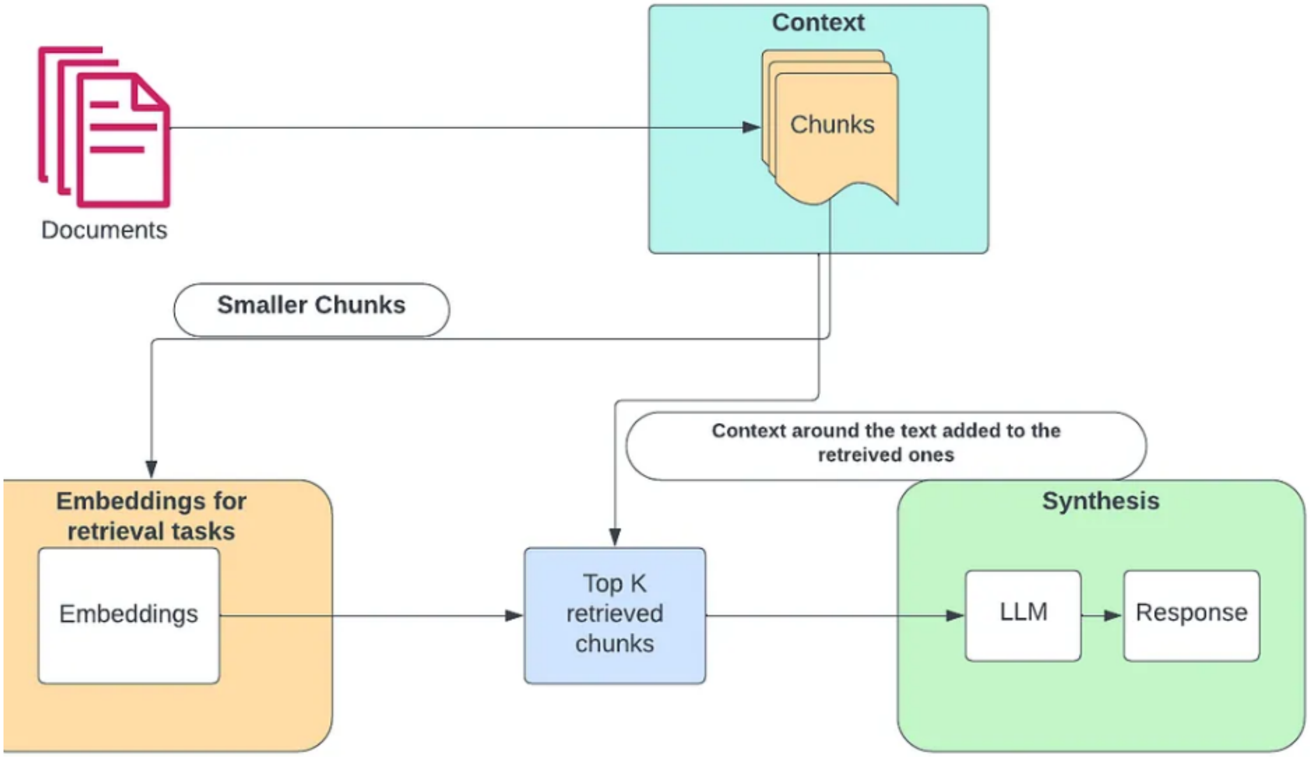
1. **Speech-to-Text**: The Whisper model transcribes voice inputs into text, enabling users to interact with the chatbot using natural speech. Its deep learning capabilities ensure accurate transcription, even for complex queries and varied accents.

2. **Text-to-Speech**: The gTTS engine converts text-based responses into speech, offering an auditory response for users who prefer listening to answers rather than reading them.

3. **Document Preprocessing and Indexing**: Text preprocessing cleanses the document data by removing non-alphanumeric characters, converting to lowercase, and eliminating stopwords. This ensures high-quality data for indexing. The document is then indexed using **Gemini Embeddings** for semantic understanding and **keyword-based indexing** for fast retrieval.

4. **Hybrid Retrieval System**: A custom retriever combines **vector-based retrieval** using Gemini Embeddings with **keyword-based matching** to optimize the relevance of responses. This hybrid system ensures that both semantically similar concepts and keyword-based matches are considered, enhancing query accuracy and response relevance.

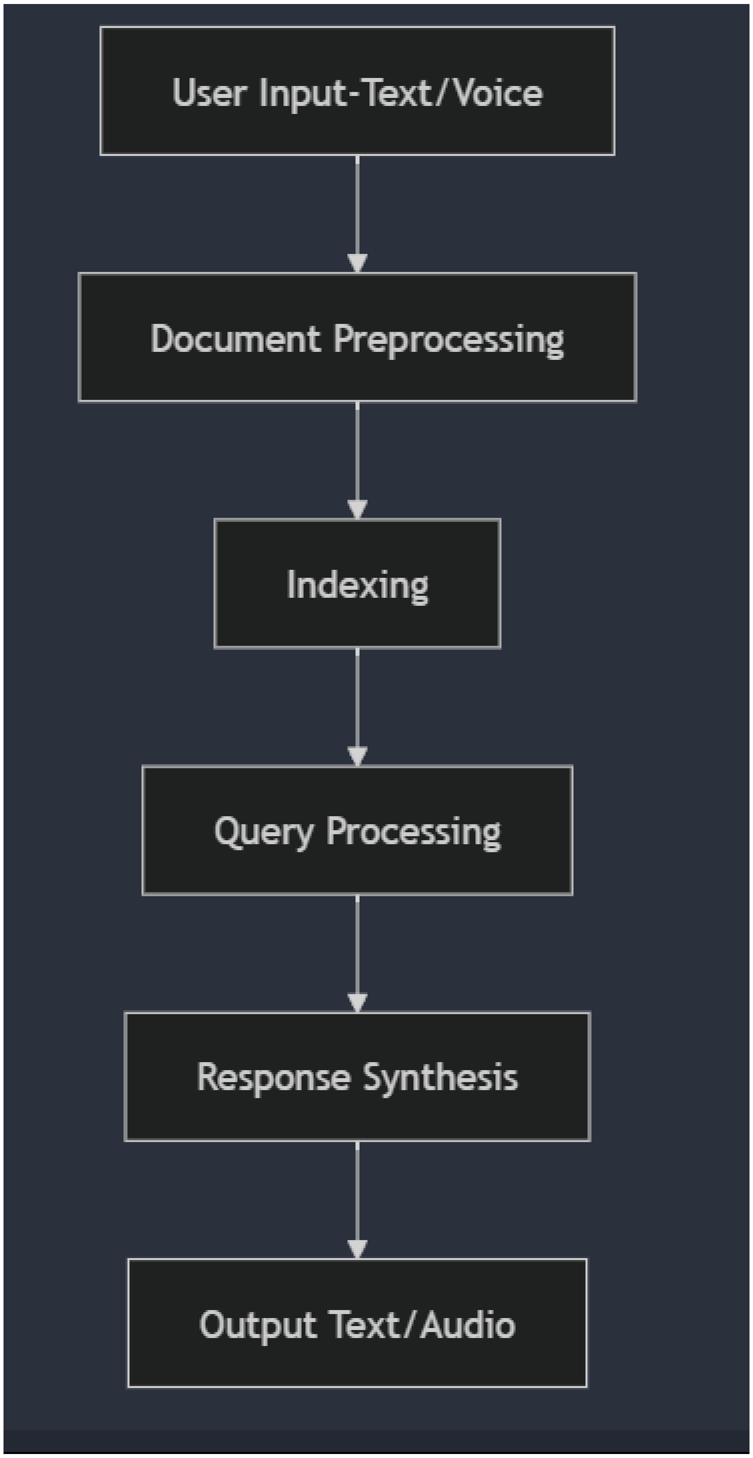
6. Methodology Adopted



The Constitution Chatbot employs a structured approach with a focus on user experience and accurate retrieval:

1. **Load and Preprocess Document Data**:
   - The chatbot uses **SimpleDirectoryReader** to load PDF documents into the system. Text preprocessing (lowercasing, punctuation removal, stopword filtering) enhances indexing quality.
2. **Create and Store Embeddings**:
   - The chatbot applies Gemini Embeddings, turning document text into vector representations that capture semantic meaning, enabling context-aware retrieval.
3. **Retrieve Information Based on User Query**:
   - A custom hybrid retriever is used to balance vector-based and keyword-based retrieval, controlled by an AND/OR logic to fine-tune response specificity.
4. **Synthesize Responses**:
   - The response synthesizer constructs coherent answers based on the retrieved information, summarizing or paraphrasing content where appropriate.

6.1 Flowchart



6.2 Hardware and Software Used

Software:

- Streamlit: Provides the user interface.
- Whisper: Handles audio-to-text transcription.
- gTTS: Converts text responses to speech.
- Gemini API: Supplies embeddings for semantic document indexing.

Hardware:

- Recommended: Modern CPU with support for efficient processing of large NLP models. For large Whisper models, a GPU can significantly improve performance.

7 Results and Discussions

The Constitution Chatbot's testing phase demonstrated a range of functional capabilities, as well as certain challenges that highlight opportunities for optimization.

Capabilities

1. **Accurate-Transcription**
   The chatbot uses Whisper for Speech-to-Text (STT) processing, which has shown to deliver high-quality transcriptions of user audio inputs. Whisper accurately captured user queries even across different accents and varied audio conditions, proving effective for real-world applications.
2. **Responsive-Text-to-Speech(TTS)_Output**
   The integration of Google Text-to-Speech (gTTS) enabled the chatbot to convert text-based responses into audio, broadening accessibility for users who prefer or require auditory feedback. The TTS system demonstrated responsiveness, providing a smooth and interactive user experience for both text and audio responses.
3. **Effective-Information-Retrieval**
   By combining vector-based retrieval using Gemini Embeddings with keyword-based search, the chatbot effectively handled nuanced and complex queries. This hybrid approach helped the chatbot return accurate responses even when the user's question did not match the exact wording found in the document. The vector-based retrieval system allowed the chatbot to interpret the context and semantic meaning of questions, while the keyword index offered fast, targeted search capability.

Challenges

1. **Latency**
   Certain processes, particularly Whisper's transcription and embedding generation, introduced latency issues. While Whisper provided high accuracy, its response time can be impacted by the size and complexity of the model used, especially in scenarios requiring rapid interactions. Similarly, embedding generation can introduce delays due to its computational demands. Future optimization could involve balancing model size with performance needs or caching frequently used embeddings to reduce response time.
2. **Memory-Management**
   Handling large documents, such as the Indian Constitution, posed memory management challenges. Processing and indexing extensive PDF files require significant memory resources. In some cases, partitioning documents into smaller sections could enhance manageability; while limiting indexing to frequently queried sections could reduce memory load without compromising on usability. This would help in maintaining the chatbot's efficiency and responsiveness when accessing extensive legal or informational texts.

8 Conclusion

The Constitution Chatbot has successfully integrated multiple AI-driven features to make document querying more accessible, especially for users seeking information in a complex text like the Indian Constitution. By combining Speech-to-Text (STT) transcription through Whisper, a Text-to-Speech (TTS) interface with gTTS, and a hybrid retrieval system using both keyword and vector-based indexing, the chatbot offers a robust, user-friendly experience.

During testing, Whisper's transcription capability proved highly accurate across diverse audio conditions, effectively capturing user queries despite variations in accent or pronunciation. The TTS component, powered by gTTS, ensured the chatbot could engage in auditory responses, broadening accessibility for users who benefit from audio feedback. The dual-index retrieval system, using both Gemini Embeddings and a keyword-based approach, improved the chatbot's response quality, allowing it to address nuanced queries that do not align exactly with the original document wording.

However, the project identified certain challenges, particularly around latency and memory management. Whisper and the embedding generation process introduced delays, especially when processing extensive documents or running on larger models. Memory management was also a challenge with large documents, such as the Constitution, where processing, indexing, and searching through dense text require significant resources. Strategies like embedding caching, document partitioning, and selective indexing could improve response time and resource management.

Future Improvements and Features

To enhance the Constitution Chatbot's usability, efficiency, and scope, the following features and optimizations could be considered:

1. Dynamic Model Scaling and Optimization

- **Latency Reduction:** By dynamically scaling the size of the Whisper and embedding models, the chatbot could adjust resource usage based on query complexity or user preference for response speed over detail.
- **On-Device Processing Options:** For devices with higher computational power, enabling on-device processing for STT and TTS could reduce dependency on cloud-based processing, thereby reducing latency.
- **Batch Embedding Generation:** Generate embeddings in batches for larger document sections or for frequently queried topics, which would improve performance for recurring user questions.

2. Enhanced Memory and Resource Management

- **Smart Caching:** Implement a caching mechanism for the most frequently accessed queries and embeddings, allowing faster retrieval without repeated processing.
- **Selective Indexing:** Introduce the option to index specific sections or clauses based on user query history, improving memory management without compromising accessibility.

3. Advanced Query Understanding

- **Synonym and Concept Recognition:** Integrate a model capable of recognizing synonyms, context, and related concepts beyond exact phrase matching. This would enhance retrieval accuracy, particularly for legal terms or uncommon phrases.
- **Follow-Up Question Handling:** Enable context-aware follow-up queries that recognize previous questions to allow for a more conversational, multi-turn dialogue with users.

4. Multi-Language Support

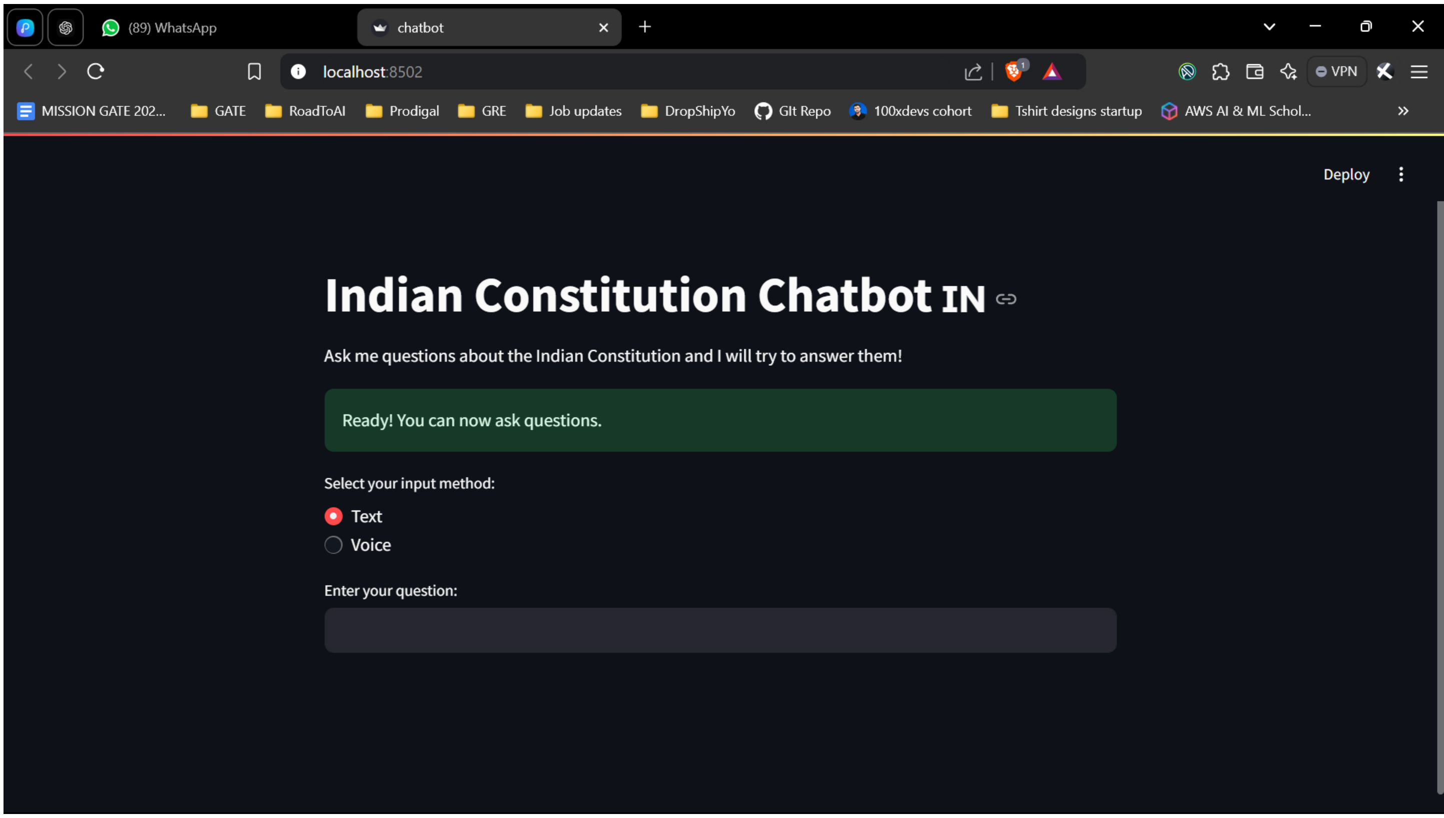
- **Additional Language Models:** Expanding the chatbot's language capabilities would make it accessible to a wider audience, particularly non-English speakers. Adding models to transcribe and interpret queries in local languages would make the chatbot more inclusive.
- **Contextual Language Switching:** For users in multilingual regions, the chatbot could recognize and switch languages seamlessly based on input, further enhancing accessibility.

5. Enhanced Accessibility Features

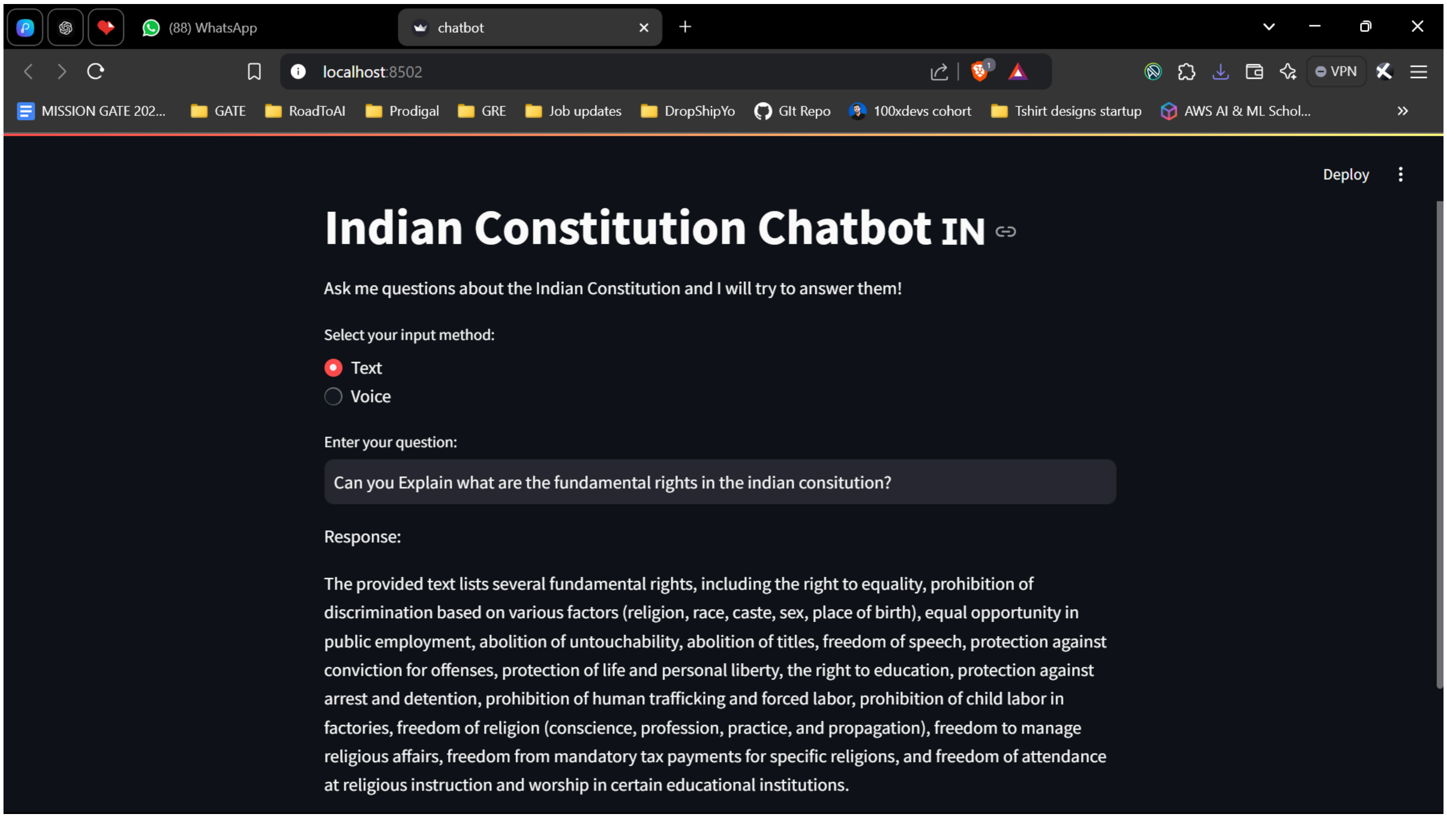
- **Adjustable Audio and Text Feedback Options:** For auditory output, offer customization in terms of speed, pitch, and accent in TTS responses. Additionally, provide adjustable font sizes and text formats for users with visual impairments.

6. Integration with Other Knowledge Bases

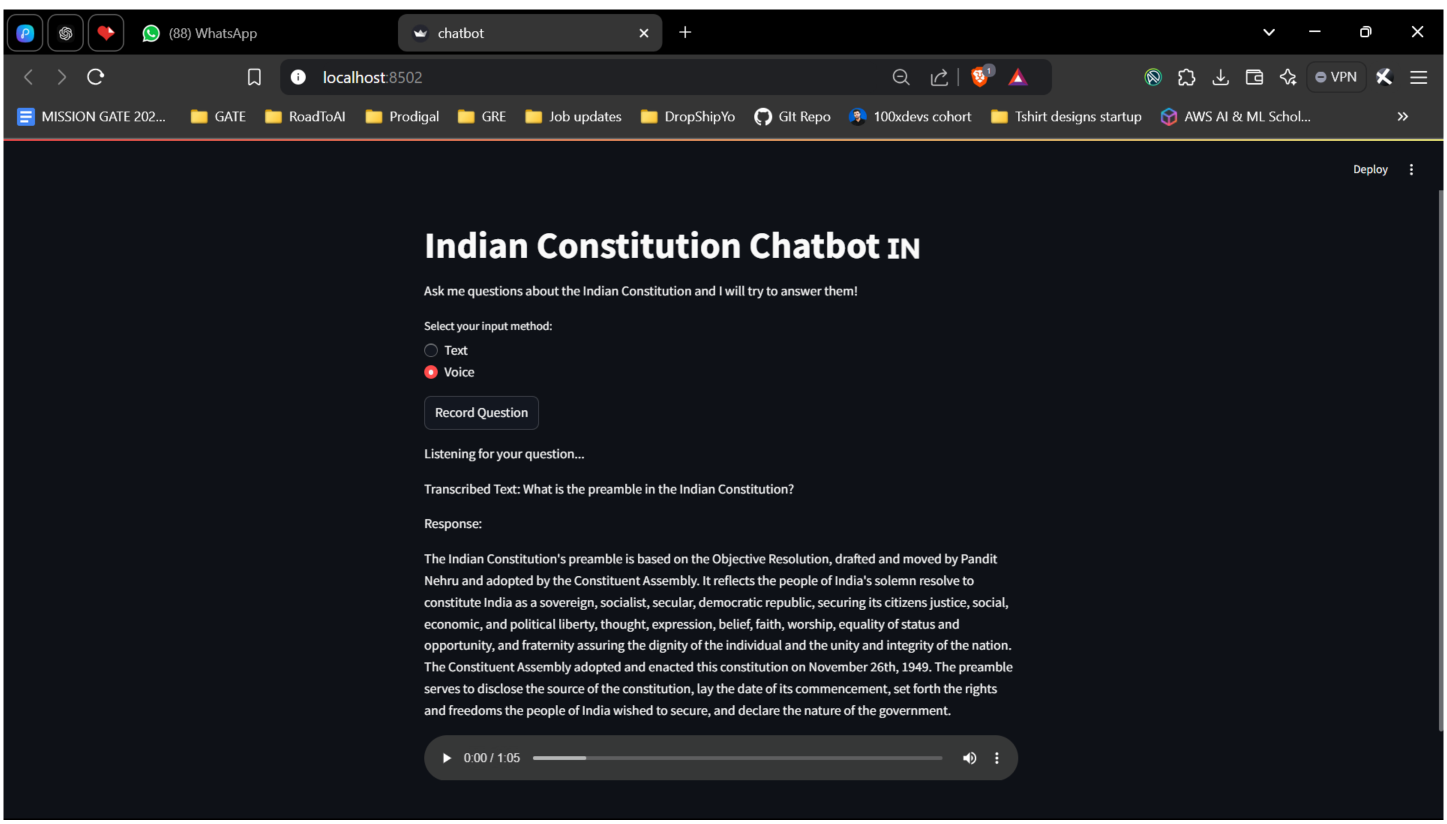
- **Supplementary Legal Documents and Case Law Integration**
- **Cross-Referencing Feature:** Implement cross-referencing between the Indian Constitution and other legal documents, highlighting related articles or sections for users with comprehensive legal queries.



Text Based Question Answering



Voice based question answering



9 References/Bibliography

- https://legislative.gov.in/constitution-of-india/
- https://aws.amazon.com/what-is/retrieval-augmented-generation/
- https://www.llamaindex.ai/blog/llamaindex-gemini-8d7c3b9ea97e
- https://medium.com/google-cloud/building-and-evaluating-basic-and-advanced-rag-applications-with-llamaindex-and-gemini-pro-in-4a3676a97e78
- https://nexocode.com/blog/posts/integrating-llms-rags-for-knowledge-base-chatbot/
- https://arxiv.org/pdf/2407.19994
- https://arxiv.org/pdf/2404.01037
- https://modulai.io/blog/rag-at-large-enterprises/
- https://www.datacamp.com/tutorial/fine-tuning-large-language-models
- https://ubiops.com/fine-tune-a-model-on-your-own-documentation/
- https://huggingface.co/mattshumer/Reflection-Llama-3.1-70B
- https://arxiv.org/pdf/2310.02226
- https://www.datacamp.com/tutorial/langgraph-tutorial
- https://www.pinecone.io/learn/vector-embeddings/
- OpenAI. *Whisper Model Documentation*. Available at: https://github.com/openai/whisper
- Google. *Google Text-to-Speech (gTTS) Documentation*. Available at: https://pypi.org/project/gTTS/
- Streamlit Inc. *Streamlit Documentation*. Available at: https://docs.streamlit.io/
- **OpenAI Whisper:** Whisper Documentation – Documentation and GitHub repository for the Whisper model, used for Speech-to-Text (STT) transcription.
- **Google Text-to-Speech (gTTS):** gTTS Documentation – Python library documentation for converting text to audio.
- **Streamlit Framework:** Streamlit Documentation – Documentation for Streamlit, a Python-based app framework for developing interactive applications.
- **Llama Index (formerly GPT Index):** Llama Index Documentation – Detailed documentation for creating, managing, and querying index structures to store and retrieve document content.