# AI-Powered Chatbot for Efficient Grievance Resolution in CPGRAMS Portal


Team Members:

**Leader**:

Name: Aditya Jadhav

Academic Year: 3rd Year (Final Year)

Institution: Government Polytechnic Nagpur

**Team Member**:

Name: Prajwal Patil

Academic Year: 3rd Year (Final Year)

Institution: Government Polytechnic Nagpur

# Abstract:

The "Grievance ChatBot" project aims to enhance the user experience for individuals navigating the CPGRAMS (Centralised Public Grievance Redress and Monitoring System) portal. Developed as an AI/ML-driven solution, the chatbot provides Ministry-specific assistance to citizens, addressing common queries related to filing grievances on the CPGRAMS portal. Leveraging state-of-the-art language models and innovative technologies, including LLM for question-answering, SentenceTransformerEmbeddings for embeddings, and FAISS for efficient vector storage, the chatbot ensures expedited and accurate responses.

# Introduction:

The Grievance ChatBot project is a cutting-edge solution designed to simplify and expedite citizen interactions with the CPGRAMS portal. The chatbot leverages state-of-the-art technologies to provide Ministry-specific assistance for common queries related to grievance filing.

This AI-driven chatbot utilizes language models Mistral 7B(Quantized) for question-answering, SentenceTransformerEmbeddings for understanding user input, and FAISS for efficient storage of information. The custom prompt template guides the chatbot's responses, ensuring accuracy and context relevance. Offensive language detection is incorporated to maintain a respectful environment.

With an intuitive Streamlit interface, users can seamlessly interact with the chatbot, asking questions about CPGRAMS or grievance filing procedures. The project not only enhances user experience but also showcases the potential of advanced technologies in public service. This documentation serves as a comprehensive guide, detailing the system architecture, implementation, and future possibilities for improvement.

# System Architecture:

The Grievance ChatBot project is built on a robust and modular architecture, integrating various components to deliver an efficient and responsive user experience. The architecture can be broken down into the following key components:

## 1) User Interface (Streamlit):

- The front-end of the application is developed using Streamlit, providing an interactive and user-friendly interface for citizens to engage with the chatbot.
- Streamlit allows for seamless integration of user input, predefined questions, and dynamic responses.

## 2) Language Models and Embeddings:

- The core of the chatbot leverages language models and embeddings for natural language understanding and generation.
- **Question-Answering Function (RetrivalQA):** This function facilitates the usage of the LLM (Language Model) for question-answering. For this project, the underlying language model is xxx (insert specific model name).
- **SentenceTransformerEmbeddings:** These embeddings enhance the model's ability to comprehend user input, contributing to accurate and context-aware responses.

## 3) Vector Storage (FAISS):

- For efficient storage and retrieval of information, the project utilizes FAISS

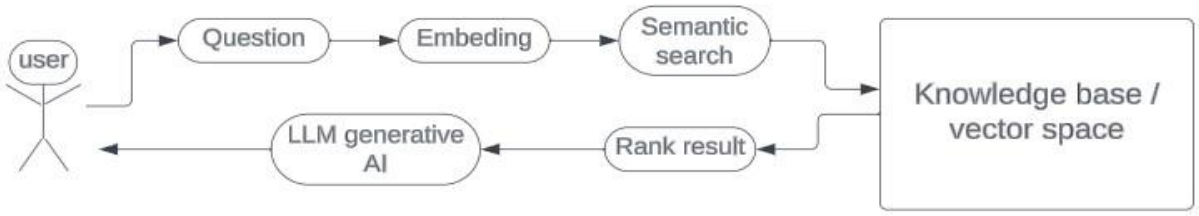


Fig.1. Program Architecture

(Vector Similarity Search), a vector storage library.

- FAISS is loaded with pre-trained embeddings, allowing the chatbot to quickly search for relevant information during user interactions.

4) Custom Prompt Template:

- A custom prompt template serves as a guide for the chatbot's responses, ensuring that answers are structured appropriately and aligned with the given context.
- The template facilitates coherent and informative responses, even in situations where the exact answer may not be present in the provided context.

5) Offensive Language Detection:

- The system incorporates a mechanism for offensive language detection to maintain a respectful and inclusive environment.
- The **better_profanity** library is utilized to identify and handle offensive content appropriately.

6) Session Management:

- Session management is implemented to keep track of the conversation history, generated responses, and user inputs.
- This ensures a coherent and context-aware conversation flow throughout the user interaction.

7) Deployment and GPU Handling:

- The system is designed to check for GPU availability during deployment, dynamically adapting to either CPU or GPU for processing.
- The deployment setup includes loading the necessary language models and embeddings onto the chosen device.

8) Main Program (Streamlit App):

- The main program, structured around the Streamlit app, orchestrates the interaction between the user interface, language models, embeddings, and vector storage.
- User queries trigger the question-answering process, and the responses are displayed on the Streamlit interface.

## Language Model (LLM - Mistral-7B-Instruct):

The Grievance ChatBot project utilizes the Mistral-7B-Instruct language model (LLM) for advanced question-answering capabilities. This model, specifically quantized as mistral-7b-instruct-v0.1.Q4_K_M.gguf, plays a pivotal role in interpreting user queries and generating accurate responses within the context of CPGRAMS.

1) Quantized Model:

The use of the quantized version of the Mistral-7B-Instruct model is a strategic choice to optimize the model's performance. Quantization involves reducing the precision of the model's weights, resulting in a more memory-efficient and faster execution without significant loss of accuracy. This is particularly crucial for real-time applications like the Grievance ChatBot, where responsiveness is key.

2) Reasons for Special Use of Mistral Model:

The selection of the Mistral-7B-Instruct model is based on its specialized capabilities in handling instructional language. Given the nature of queries related to grievance filing and CPGRAMS procedures, a model fine-tuned for instructive contexts aligns seamlessly with the project's objectives. This model excels in comprehending and generating responses in instructional and informational scenarios, making it an ideal choice for the chatbot's domain.

3) Model Requirements:

| Name | Quant method | Bits | Size | Max RAM required | Use case |
|---|---|---|---|---|---|
| mistral-7b-instruct-v0.2.Q4_K_S.gguf | Q4_K_S | 4 | 4.14 | 6.64 GB | small, greater quality loss |

Fig2. Mistral 7B requirements

4) Model Comparison Chart:

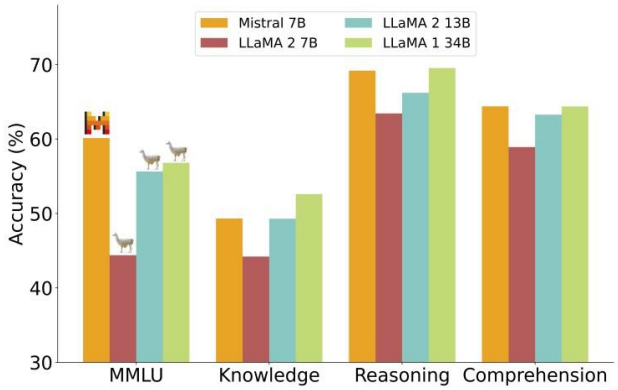


Fig3. Mistral model Accuracy comparision

5) Quantization Benefits:

Quantizing the Mistral-7B-Instruct model brings several advantages:

- Memory Efficiency: The reduced precision of weights leads to lower memory usage, enabling smoother deployment and execution.
- Faster Inference: Quantization enhances inference speed, contributing to a more responsive user experience in real-time interactions.

## Dataset and Vector Creation:

In crafting the Grievance ChatBot, a purposeful dataset is assembled from CPGRAMS, encompassing diverse user queries and responses. This dataset forms the basis for creating embeddings and building an efficient vector store for streamlined information retrieval.

- Data Collection: Meticulously curated from CPGRAMS, ensuring a representative sample of user interactions.
- Embeddings and Vectorization: Utilizing SentenceTransformerEmbeddings, textual data is transformed into semantic vectors, facilitating effective comprehension by the chatbot.
- Vector Storage (FAISS): FAISS is employed for storing and organizing the vectorized data, providing a fast and precise retrieval mechanism during user interactions.

## Potential Enhancements:

1) Expanded Knowledge Base:
Increase the dataset size to cover a broader range of user scenarios and grievances, enhancing the chatbot's knowledge base.

2) Contextual Understanding:
Fine-tune the language model for even better contextual understanding, allowing the chatbot to provide more nuanced and precise responses.

3) Multi-lingual Support:
Extend language support to cater to a wider audience, ensuring inclusivity for users who communicate in different languages.

4) Real-time Updates:
Implement mechanisms to update the dataset in real-time, ensuring the chatbot stays current with the latest information and trends.

5) User Feedback Integration:
Incorporate a feedback mechanism for users to provide insights, enabling continuous improvement and addressing evolving user needs.

6) Enhanced User Interface:
Improve the Streamlit interface for a more intuitive user experience, making interaction with the chatbot even more user-friendly.

7) Adaptive Learning:
Implement adaptive learning mechanisms to enable the chatbot to learn from user interactions over time, refining its responses based on feedback.

# References:


1. Langchain Documentation - Getting Started: https://python.langchain.com/docs/get_started/introduction](https://python.langchain.com/docs/get_started/introduction

2. Langchain Documentation - Vector Stores with FAISS: https://python.langchain.com/docs/integrations/vectorstores/faiss](https://python.langchain.com/docs/integrations/vectorstores/faiss

3. Mistral-7B-Instruct Model Paper: https://extension://ngphehpfehdmjellohmlojkplilekadg/pages/pdf/web/viewer.html?file=https%3A%2F%2Fmistral.ai%2Fassets%2FMistral_7B_paper_v_0_1.pdf

4. Anaconda Blog - How to Build a Retrieval-Augmented Generation Chatbot: https://www.anaconda.com/blog/how-to-build-a-retrieval-augmented-generation-chatbot