**GenAI Chatbot Project**

Approach

## Data Processing

The first step in building the chatbot was to retrieve and structure relevant data from GitLab’s Handbook and Direction pages. This was achieved using web scraping techniques with the requests and BeautifulSoup libraries. The fetch\_data function in data\_processing.py handles this task.

This function takes a URL as input, fetches the content of the page, and extracts the text using BeautifulSoup. The extracted text is then used as the context for the chatbot.

## Chatbot Implementation

The core functionality of the chatbot is implemented using the transformers library from Hugging Face. The GenAIChatbot class in chatbot.py uses a pre-trained model to answer questions based on the provided context.

The get\_answer method takes a question and context as input and returns the answer generated by the model. If the question or context is empty, appropriate messages are returned.

## Frontend/UI Development

The user interface for the chatbot is developed using Streamlit, a popular framework for building interactive web applications. The UI code is located in app.py.

This code sets up the Streamlit app, initializes the chatbot, fetches data from GitLab’s Handbook and Direction pages, and handles user input to generate responses. The chat history is maintained using Streamlit's session state, and options are provided to clear, save, and restore chat history.

## Unit Tests

Unit tests are implemented in test\_chatbot.py to ensure the chatbot functions correctly.

These tests cover basic functionality, such as answering questions, handling empty questions, and following up on previous questions.

# Key Technical Choices

1. **Data Processing with BeautifulSoup**: Chosen for its simplicity and effectiveness in extracting text from HTML content.
2. **Transformers Library**: Utilized for its state-of-the-art pre-trained models, which provide high-quality answers to user queries.
3. **Streamlit for UI**: Selected for its ease of use and ability to quickly create interactive web applications.
4. **Unit Testing with Unittest**: Ensures the chatbot functions correctly and handles various scenarios.

# Completed Tasks

## Project Documentation:

The README.md file has been created and includes instructions for setting up and running the chatbot locally.

## GitHub Repository

The repository contains all source code and files, including a README.md with detailed setup instructions.

## Chatbot Core Functionality

* Data Processing: Implemented in data\_processing.py
* Chatbot Implementation: Implemented in chatbot.py.
* Frontend/UI Development: Implemented using Streamlit in app.py.
* Unit Tests: Implemented in test\_chatbot.py.

# Pending Tasks

### Public Deployment

The chatbot needs to be deployed on a platform like Vercel, Hugging Face Spaces, or Streamlit Community Cloud. Provide a public URL for access.

### Enhance Chatbot Functionality

Implement advanced or creative features beyond the basic requirements. Add innovative guardrailing and transparency features. Focus on product thinking to enhance the user experience, especially for employees.

### Gather Feedback and Iterate

Collect feedback from users . Use the feedback to make iterative improvements to the chatbot. Ensure the chatbot remains relevant and useful for its intended audience.

By following this plan, we can ensure the successful completion and deployment of the GenAI Chatbot Project, providing a valuable tool for users to access information from GitLab’s Handbook and Direction pages in an engaging and accessible way.