Algorithm and Dataset

Description of Project:

Create a simple chatbot that can answer predefined questions using rule-based logic.

Algorithm:

The algorithm for this rule-based chatbot consists of several defined steps, inputs, outputs, conditions, loops, and required libraries. Below is a detailed description of the algorithm:

1. Load Environment Variables:

- **Step:** Use the dotenv library to load environment variables from a .env file.
- **Input:** .env file containing sensitive information such as API keys.
- **Output:** Loaded environment variables.
- **Conditions:** The .env file must be present and contain the required variables.
- **Libraries:** dotenv, os.

2. Configure Streamlit Page Settings:

- **Step:** Set up the Streamlit page with specific settings like title, favicon, and layout.
- **Input:** Configuration settings for the page.
- **Output:** Configured Streamlit page.
- **Conditions:** None.
- **Libraries:** streamlit.

3. Configure Google Gemini-Pro AI Model:

- **Step:** Initialize and configure the Google Gemini-Pro AI model with the API key.
- **Input:** API key from environment variables.
- **Output:** Configured AI model ready to process user queries.
- **Conditions:** The API key must be valid.
- **Libraries:** google.generativeai.

4. Initialize Chat Session:

- **Step:** Start a new chat session if one does not already exist in the session state.
- **Input:** Existing session state.
- **Output:** Initialized chat session.
- **Conditions:** Check if a session already exists.
- **Libraries:** streamlit.

5. Role Translation Function:

- **Step:** Translate roles between Gemini-Pro and Streamlit terminology.
- **Input:** User roles (e.g., "model", "user").
- **Output:** Translated roles (e.g., "assistant", "user").
- **Conditions:** Roles must be defined.
- **Libraries:** None.

6. Display Chat History:

- **Step:** Iterate through the chat history and display each message.
- **Input:** Chat history from the session state.
- **Output:** Displayed chat messages on the Streamlit page.
- **Conditions:** There must be a chat history.
- **Libraries:** streamlit.

7. User Input Handling:

- **Step:** Capture user input through the chat input field, process it, and send it to the AI model.
- **Input:** User's message.
- **Output:** Response from the AI model, displayed on the page.
- **Conditions:** User input must be valid and non-empty.
- **Libraries:** streamlit, google.generativeai.

8. Generate Response:

- **Step:** Send the user's message to the Gemini-Pro AI model and receive a response.
- **Input:** User's message.
- **Output:** AI-generated response.
- **Conditions:** The AI model must be correctly configured and accessible.
- **Libraries:** google.generativeai.

9. Display Response:

- **Step:** Display the AI model's response in the chat interface.
- **Input:** AI-generated response.
- **Output:** Displayed response message on the Streamlit page.
- **Conditions:** The response must be valid.
- **Libraries:** streamlit.

Dataset:

This project does not use a traditional dataset as it relies on predefined questions and answers processed by the Google Gemini-Pro AI model. The predefined questions and answers act as the knowledge base for the chatbot. Here are the key elements:

1. Predefined Questions and Answers:

- **Source:** Manually created or sourced from common FAQs.
- **Format:** Simple pairs of questions and their corresponding answers.
- **Usage:** These pairs are used to form the rule-based logic that the chatbot follows.

2. AI Model Configuration:

- **Source:** Google Gemini-Pro AI.
- **Format:** Pre-trained model capable of generating responses based on user input.
- **Usage:** The model processes user queries and generates appropriate responses based on its training and the predefined rules set by the developer.

3. Environment Variables:

- **Source:** .env file.
- **Format:** Key-value pairs (e.g., GOOGLE_API_KEY=your_api_key).
- **Usage:** Used to securely store and access sensitive information required for API configuration.

Libraries Used:

1. **Streamlit:** For creating the web interface.
2. **Google Generative AI:** For configuring and using the Google Gemini-Pro AI model.
3. **dotenv:** For loading environment variables.
4. **os:** For accessing environment variables.

Summary of Steps and Flow:

1. Load environment variables using dotenv and configure the API key.
2. Set up the Streamlit page with the desired configuration.
3. Initialize and configure the Google Gemini-Pro AI model.
4. Start a chat session in Streamlit, maintaining chat history.
5. Implement a role translation function to map roles between Gemini-Pro and Streamlit.
6. Display the chat history on the Streamlit page.
7. Capture and process user input, sending it to the AI model.
8. Receive and display the AI model's response.

By following these steps, the chatbot can efficiently answer predefined questions using rule-based logic, providing users with quick and accurate information. The integration of Streamlit and the Google Gemini-Pro AI model ensures a seamless and interactive user experience.