E-Commerce Chatbot Functionalities

The **E-Commerce Chatbot** is a comprehensive solution designed to assist users with queries related to products and orders within an e-commerce platform. Below is a detailed overview of the functionalities implemented in the project, derived directly from the provided codebase.

1. Product Information Retrieval

- **Product Queries**: Allows users to inquire about various products available in the store. Users can obtain detailed information such as product titles, descriptions, features, ratings, prices, and categories.
- **Retrieval-Augmented Generation (RAG)**: Enhances the accuracy and relevance of product-related responses by retrieving pertinent information from the product dataset before generating a comprehensive answer using language models.
- **Top-K Results**: Users can specify the number of top relevant products they wish to view based on their query, ensuring they receive the most pertinent information.

2. Order Information Access

- **Order Queries via Customer ID**: Enables users to fetch detailed order histories by providing their unique Customer ID. This includes information like order dates, product categories, sales amounts, shipping costs, payment methods, and order priorities.
- **Conversational Order Assistance**: Facilitates interactive conversations where users can ask follow-up questions about their orders. The chatbot maintains conversation history to provide contextually accurate responses.
- **Mock API Integration**: Utilizes mock API endpoints to simulate interactions with order data, ensuring seamless retrieval and processing of order-related information.

3. Chatbot Interaction

- **General Chatbot Queries**: Supports a wide range of user queries beyond just products and orders, allowing for a more natural and engaging conversational experience.
- **Contextual Responses**: Leverages conversation history and retrieved data to generate responses that are both relevant and context-aware, enhancing user satisfaction.
- **Error Handling**: Implements robust error handling to manage invalid inputs or unexpected issues gracefully, ensuring a smooth user experience.

4. Health Monitoring

- **Healthcheck Endpoint**: Provides a simple endpoint to verify the operational status of the chatbot API. This is essential for monitoring and ensuring the reliability of the service.

5. Data Processing and Management

- **Data Preprocessing**: Handles the cleaning and preparation of raw datasets for products and orders. This includes managing missing values, normalizing data, and structuring information for efficient retrieval.
- **Data Models**: Defines structured data models using Pydantic to validate and serialize product and order information, ensuring data integrity and consistency across the application.

6. Retrieval-Augmented Generation (RAG) Service

- **Document Embedding**: Uses Sentence Transformers to generate embeddings for product descriptions, enabling efficient similarity calculations and information retrieval.
- **Similarity Scoring**: Employs cosine similarity to identify and retrieve the most relevant documents in response to user queries, ensuring high-quality and accurate responses.

7. Gemini API Integration

- **Response Generation**: Integrates with the Gemini API to generate natural language responses based on retrieved context, enhancing the chatbot's ability to provide detailed and human-like answers.
- **Prompt Engineering**: Constructs well-formatted prompts that combine user queries with relevant context, optimizing the quality of the generated responses.

8. User Interfaces

- **Streamlit Applications**: Offers user-friendly web interfaces built with Streamlit for interacting with the chatbot. There are separate interfaces for handling product-related queries and order-related inquiries, providing intuitive ways for users to engage with the chatbot.
    - **Orders Chatbot Interface**: Allows users to input their Customer ID and pose questions about their order history.
    - **Products Chatbot Interface**: Enables users to search for products and receive detailed information based on their queries.

9. Logging and Monitoring

- **Custom Logging Setup**: Implements a logging system to track application behavior, monitor performance, and facilitate debugging. Logs include information about API requests, responses, errors, and other critical events.

10. Testing

- **Unit and Integration Tests**: Includes a comprehensive suite of tests to verify the functionality of individual components and their interactions. This ensures the reliability and robustness of the chatbot across various scenarios.

11. Configuration Management

- **Environment Variables**: Utilizes environment variables to manage sensitive information like API keys and file paths, enhancing security and flexibility in different deployment environments.
- **Configuration Files**: Centralizes configuration settings, making it easier to manage and update parameters across the application.