Overall Approach

The approach involves using a Flask web application to serve a chatbot interface. The chatbot uses SentenceTransformers to find the best match for user queries from a given corpus and sample question-answer pairs. The conversation history is maintained using Flask sessions to handle follow-up questions effectively.

Frameworks/Libraries/Tools Used

- Flask: For creating the web application and handling HTTP requests.
- SentenceTransformers: For encoding sentences and finding the best match for user queries.
- util (from SentenceTransformers): For computing cosine similarity between query and corpus embeddings.
- json: For reading and parsing the sample question-answer file.
- os: For handling file paths.

Problems Faced and Solutions

1. Corpus and Sample Q&A Integration:

- Problem: Ensuring the chatbot answers accurately using the given corpus and sample question-answer pairs.
- Solution: Used SentenceTransformers to encode the questions and find the best match for user queries, ensuring the chatbot provides relevant answers.

2. Maintaining Conversation History:

- Problem: Handling follow-up questions by maintaining context.
- Solution: Used Flask sessions to keep track of conversation history and provide relevant responses based on previous interactions.

3. Performance and Latency:

- Problem: Reducing response time to within acceptable limits.
- Solution: Optimised model usage and response generation to ensure minimal latency, aiming for response times within 2-3 seconds.

Future Scope

1. Enhanced Context Handling:

- Improve the conversation history management to better understand and respond to complex queries that reference earlier parts of the conversation.

2. Personalization:

- Implement user-specific personalization, allowing the chatbot to remember user preferences and provide tailored recommendations.

3. Multilingual Support:

- Extend support for multiple languages to cater to a broader audience.

4. Voice Interaction:

- Add voice interaction capabilities, allowing users to interact with the chatbot using speech.

5. Integration with External APIs:

- Integrate with external wine databases and APIs to provide real-time information and recommendations.

By implementing these improvements, the chatbot can become more user-friendly and provide a richer experience for users seeking information about wines.