# JESSUP CELLARS CHATBOT

1. Overall approach:
    a. I have used the Gemini 1.5 pro model for the chatbot. I trained the model on Google AI Studio from the provided corpus and sample q&a.
    b. I used Google Generative AI SDK to interact with the Gemini model.
    c. Next, I configured the model to respond to queries based on the provided corpus and sample questions and answers.
    d. Users call the API to the Gemini model whenever they ask any query. The API call to the Gemini model is made via the API key.
    e. I also managed a chat session to maintain conversation history and context.
    f. Furthermore, I implemented logic to provide relevant answers from the corpus and direct users to contact the business for out-of-scope queries.

2. Frameworks/Libraries/Tools Used:
    a. Node.js: Primary programming environment,  also to host the chatbot on localhost.
    b. Google Generative AI SDK: For accessing and interacting with the Gemini model.
    c. Environment Variables: I used ‘process.env.API_KEY’ to securely manage API keys.

3. Problems:
    a. The POST and GET methods were not working properly initially. Only a blank UI of the chatbot would be visible to users. I fixed this by revising the code and making proper routes.
    b. I encountered a few problems with the API key, where the API key was inaccessible despite being set-up properly and within the token limit. For any query, the output would be “undefined”. This was fixed a few minutes after I created the API key.

4. Future Scope:
    a. Faster input processing and analysis, response fetching and output generation.
    b. Multilingual support.
    c. Use analytics to track chatbot’s performance to allow continuous improvement.
    d. Integrating it within a real, fully-functioning website.