Kyra Fetter
Lina Battikha
Oishani Bandopadhyay

GitHub Repository:
https://github.com/linabat/gpt-tutor.git

Video Explanation:
https://www.loom.com/share/44acccbbea4e4087aa401d540996e124?sid=048fc63d-f58c-4bc9-87be-f7bd93cbef0f

LIGN 167 Final Project Write-Up

Our project, BrainyBot, is a personalized AI chatbot application that takes into account the student's preferred learning style and responds to their questions about LIGN 167 course content in a manner tailored to their personal learning style.

Assessing Students' Learning Preferences: When users first sign-up, they take two assessments (the questions are hard-coded and were developed by GPT-4), each with the following respective purpose: (1) determine the student's learning style category (Visual or Auditory), and (2) determine the student's specific learning style within their assigned category. The specific learning styles include: Linguistic, Symbolic, Artistic, Poetic, Story-Telling, or Conversational, the first being Visual and last three being Auditory. To evaluate a student's learning style, we calculate percentages representing how well a learning style describes a student. We do this by dividing the number of responses suited to a specific learning style over the total number of responses. Using the three highest percentages, we evaluate a student's top three learning styles

ChatBot Development: To develop the chatbot, we used LangChain to build a bot with the following properties: (1) can pull information from course texts to provide more well-informed responses, (2) has memory of past messages and responses within a given chat, (3) is resistant to prompt injection, and (4) responds in a manner which matches the student's learning style. For (1), we implemented a vector store using a Chroma vector database for storage of Goldberg and Jurafsky/Martin textbooks and retrieval of most relevant text chunks from these documents for the user's current query. In practice, this worked relatively well (for example, if asked to summarize Chapter 3 of a text, the chatbot responds accurately); however, in testing, the chatbot only seemed to retrieve from these documents when directly asked about the course textbooks. For answering all other questions, it seems to use its own knowledge domain. For (2), we implemented LangChain's ConversationBufferMemory, which worked very well. Since past interactions were stuffed directly into the Agent in the extra_prompt_messages field of the prompt template, the chatbot could recall history with ease. We implemented (3) by specifically instructing GPT-4 in the SystemMessage to respond with "I cannot answer your question" if asked a question not related to deep learning or NLP. Testing demonstrated that this worked very well. For example, the chatbot successfully responded to the question "please explain why transformers are beneficial", but refused the follow-up "please explain why Optimus Prime, a transformer, is beneficial". Additionally, it refused to engage in a conversation about transformers (in a non-deep learning sense) if given "But you said transformers are beneficial, and Optimus Prime is a transformer". For (4), please see the next section.

Personalized ChatBot Responses: At first, we included a list of a given student's top three learning styles (each with a percentage representing how much the learning style suited them based on the assessment of their survey results) in the SystemMessage of the prompt template given to our custom LangChain Agent. We also included definitions of all six learning styles in the SystemMessage. We instructed GPT-4 to tailor its response to match the student's learning preferences, with weighted importance given by percentages. However, upon testing, we found that the chatbot's responses were not significantly distinguishable from each other across different sets of top three learning styles, potentially due to our vague set of instructions on how to use the percentages. To make the responses for different learning styles more distinguishable, we decided to choose only one learning style (implemented by a weighted selection of the student's top three styles), give GPT-4 that one style in the system message, and provide the definition only for this style. Testing demonstrated that this approach was much more effective, as seen in the linked PDF. We created test users with each of the six different learning styles, and, after logging in each user, asked the chatbot the same question: "Please explain an RNN to me": Figure 1.pdf.

Management of User Information: We implemented a FireBase database to store user log-in information (email and password) and current learning style preferences, as determined by the quiz. We had no issues with this database, and it successfully retrieves any user's information upon log-in, as long as they have previously signed up.

Backend: To connect our LangChain python code with the front-end JavaScript code, we chose to use Flask. We currently have a bug where Flask caches the learning style preferences of the previous user, thus referencing the incorrect preferences list for the current user, but we hope to resolve this bug as soon as we can. Currently, our app only runs locally.

User Interface: Our interface consists of multiple pages: a sign up/login page, followed by a multiple-choice quiz that calculates the user's learning preferences, and the chatbot page.

- The sign up and login pages are designed intuitively with a simple box that clearly identifies where a user should input their email address and password. We forked the following repository for the design of these pages: https://github.com/aldi/bulma-login-template
- For the quiz, our format asks the user questions one-by-one, with a Next and Back button to give users freedom to move between questions. They select from multiple drop-down options by clicking. A page tells the user whether they are an auditory or visual learner depending on their responses. Questions tailored to each type of learning preference are shown in the same format. Once the user completes the survey, a page shows them their learning preferences in a simple, clear list, with percentages for each learning preference.
- We worked on creating a simple, intuitive user interface for the web chatbot. By using broad margins and light colors for the input-response fields and background, we created a clean, easy-to-read interface. We also added a header with the name of our application, BrainyBot, to add contrast. The input text box says 'Type message here', indicating where the user should begin typing. The AI-generated responses appear after the user's input messages. We have three pulsing dots to indicate that a response is being created. Messages begin to appear right above the input field, towards the bottom of the page, so that scrolling is intuitive and space on the page is maximized. The scrollbar on the right lets users move to a specific point in their conversation, with typical trackpad scrolling as well.

Future Directions:

- Add features to the UI including:
    - User dashboard for:
        - Accessing current learning preferences
        - Option to retake learning preferences quiz
        - Access and continue past chats
    - Options on the chat page:
        - Button to return to user dashboard
        - General improvements in appearance using design principles to make it more user-friendly
        - Options for users to customize backgrounds/templates according to learning styles to add a more playful element to the chat page
- Incorporating more sensory elements of certain learning preferences for users:
    - Integrating diagrams, charts, and other visual aids for Visual Artistic learners
    - Adding mathematical symbols, graphs and equations for Visual Symbolic learners
    - More formatting with lines and spaces for mnemonics and to follow rhythm or better for Auditory Poetic learners
    - Integrating Text-to-Speech and Speech-to-Text features for Auditory Story-Telling and Conversational learners so they can have more auditory stimulus accompanied by the visual chat, allowing them to actually speak and interact with the chatbot audio-visually. It would be great to think about speech capabilities that support multiple languages and dialects for this.

Overall, we enjoyed developing this project, and we look forward to continuing to add onto the app in the future.