DEVELOPMENT PROCESS:

Backend Development (VSCode):

1. \*\*Package and Dependency Import:\*\*

- Open Visual Studio Code (VSCode).

- Import necessary Python packages and dependencies using the `import` statements.

- Ensure that the required libraries, such as `openai`, `streamlit`, `dotenv`, and others, are installed in the Python environment.

2. \*\*Prompt Generation for Quiz Questions:\*\*

- Utilize the `openai` library to generate quiz questions dynamically.

- Write prompts that align with the assignment's requirements for creating Multiple Choice Questions (MCQs).

- Ensure the generated questions follow the specified format, including the question, options (a, b, c, d), and correct answer.

3. \*\*User Input Integration:\*\*

- Implement functionality to take user input in the backend.

- Use the input received from the user to interact with the quiz bot.

- Handle the responses from the quiz bot accordingly, progressing through the quiz without revealing correct answers until the end.

Frontend Integration (Streamlit):

1. \*\*Initialize Chat History:\*\*

- In the Streamlit application, create a chat history to maintain the conversation flow between the user and the quiz bot.

- Ensure the chat history structure includes roles ("user," "assistant") and message content.

2. \*\*User Interaction:\*\*

- Set up the user interface in Streamlit to receive input from the user.

- Display messages from the chat history on the application to provide a conversational experience.

3. \*\*Chat Input and Output:\*\*

- Capture user input using Streamlit's `st.chat\_input` functionality.

- Display the user's input as part of the chat conversation.

- Utilize the `openai` API to get responses from the quiz bot based on the user's input.

4. \*\*Simulated Typing Animation:\*\*

- Simulate a typing animation in the frontend to enhance the user experience.

- Use Streamlit's `st.markdown` to dynamically update the displayed content, creating the effect of the bot typing the response.

5. \*\*Update Chat History:\*\*

- Keep the chat history updated with each user input and the corresponding bot responses.

- Display the entire chat history to the user for a seamless and interactive quiz experience.

## ADDITIONAL CONSIDERATIONS:

- \*\*Environment Setup:\*\*

- Ensure that the required API keys, such as OpenAI, are correctly set up in the environment.

- Verify that the Streamlit application is configured to run smoothly.

- \*\*Testing:\*\*

- Conduct thorough testing of both the backend and frontend functionalities.

- Address any issues or bugs that arise during the testing phase.

- \*\*Documentation:\*\*

- Maintain clear and comprehensive documentation for both backend and frontend code.

- Include information on how to run the application and any dependencies required.

By following this development process, you can create a fully functional quiz application with a conversational interface.

Screenshot:



A black background with white text

Description automatically generated  
A screenshot of a computer

Description automatically generated