**ASSIGNMENT**

qAIz - AI Powered Quiz Generator

Submitted by:

Abirami Sozharajan

**Introduction**

**Overview:**

A sophisticated quiz software called qAIz was developed with attention to detail to provide users with a fun and customized quiz experience. This cutting-edge platform makes use of Streamlit, Python, and Langchain to create an AI-powered quiz generator that allows users to engage in interactive quizzes on any topic they want.

**Features:**

AI-Infused Question Generation: Utilizing the latest developments in Langchain and OpenAI Chat Completion API, qAIz provides quiz questions that are highly relevant and varied on the go.

Personalized Topics: Users have the freedom to choose their preferred quiz topics, guaranteeing a customized and valuable learning experience.

**Comprehensive Question Structure:**

Four-Answer Minimum: Each quiz question presents users with a minimum of four answer options, fostering a challenging and educational quiz environment.

**Immediate Feedback:**

Real-Time Scoring: Upon submission of the quiz, users receive immediate feedback on their performance, including their score.

Correct Answer Identification: Users can identify the correct answers, enhancing the learning experience and providing valuable insights.

**Design and Implementation**

In developing the MCQ Quiz application, the primary focus was on creating an interactive and engaging experience for users who wish to participate in quizzes customized to their preferred topics. The implementation involves a two-tiered approach, emphasizing the design of the backend logic and the creation of a user-friendly front-end interface.

* **Utilizing Python for Robust Logic:**

Python is used to create the basic application logic, providing a solid basis for the quiz-generating process. Python is an ideal choice for managing the backend capabilities because of its adaptability and simplicity of integration.

* **Topic-Based Question Generation:**

The system is designed to dynamically generate questions and answers based on the user's selected topic of interest. This personalized approach ensures that the quizzes align with the user's knowledge preferences, enhancing their engagement and learning experience.

* **Comprehensive Logic Coverage:**

The implemented logic covers various aspects, including question generation, multiple answer options, and accurate identification of the correct answer. This comprehensive coverage ensures the quiz's educational value and challenges participants effectively.

* **User-Friendly Streamlit Interface:**

The front-end of the quiz application is developed using Streamlit, a powerful and intuitive framework for creating web applications with minimal effort. Streamlit allows for rapid development, providing an appealing and user-friendly interface.

* **Feature-Rich User Input:**

Users are granted the ability to input their preferred quiz topics and specify the number of questions they wish to answer. This feature enhances user customization, enabling them to customize the quiz experience to their preferrence.

* **Structured Quiz Presentation and Answer Submission:**

The quiz presentation is carefully designed, presenting users with sets of questions and answer options. Each question features a minimum of four answers, upon answering all questions, users can submit the quiz.

**Application Logic**

The **main** function serves as the entry point for the Streamlit application. It begins by setting the background color of the application to a warm tone, creating an aesthetically pleasing user interface. The PwC logo is then displayed to provide branding.

Users are prompted to input the quiz topic and specify the number of questions they desire. This information is collected using Streamlit's text input and number input components. Subsequently, the **generate\_questions** function is called to dynamically create quiz questions, answer options, and correct answers based on the user's input.

The application then enters a loop to present each generated question to the user. For each question, the question itself is displayed, and the user is prompted to select the correct answer from a set of multiple choices. The selected answer and the correct answer are displayed for each question, allowing users to track their responses.

A "**Submit"** button is provided for users to finalize their answers and trigger a rerun of the application, facilitating multiple quiz attempts. The logic of the quiz generation, presentation, and feedback is encapsulated within this main function.

The generate\_questions function plays a crucial role in dynamically generating quiz content. It iterates over the specified number of questions, sending prompts to the OpenAI API to generate both the question and the correct answer. Additionally, it generates multiple-choice options for each question by randomly selecting from a set of possible answers.

In summary, the application logic revolves around dynamically generating quiz content, presenting it to the user through a Streamlit interface, collecting user responses, and providing feedback. The integration of OpenAI's API enhances the educational value of the quiz by dynamically generating contextually relevant questions.

**qAIz User Guide**

Getting Started:

1. Open the Application: Execute the script containing the qAIz code in a Python environment using the code ‘streamlit run qAIz.py’ on a terminal window opened to the directory containing the files .The Streamlit application will launch in your default web browser.
2. Taking a Quiz: Enter your Quiz Details. Input the topic for the quiz in the text box provided. Specify the number of questions you want to answer using the number input field.
3. Generated Quiz Questions: After entering the quiz details, a set of multiple-choice questions will be dynamically generated based on the specified topic. Each question will be displayed on the screen along with multiple answer choices.
4. Answering Questions: For each question, select the correct answer by clicking on the radio button next to your choice. Your chosen answer and the correct answer will be displayed for each question.
5. Submit Your Answers: Once you have answered all the questions, click the "Submit" button at the end of the page. This will finalize your quiz attempt and trigger a rerun of the application.
6. Reviewing Results: After submitting, the application will display the questions you answered, the options you chose, and the correct answers. This allows you to review your quiz attempt and identify areas for improvement.
7. Customization: Changing Quiz Parameters- If you want to take another quiz with different parameters, you can simply modify the quiz topic and the number of questions. The application will dynamically generate a new set of questions based on your input.

**GitHub repository**: <https://github.com/Abirajan234/Pwc-Assignment.git>