**Task 3- Overview**

This task involves integrating OpenAI API calls into **mcq.py** to generate 5 challenging multiple-choice questions on topic chosen in dropdown, complexity level from radio buttons and submit the questions and generate the feedback for all questions.

**Task List**

1. **Understand the Boilerplate Code**: Review the structure and logic for multiple-choice question (MCQ) generation and evaluation.
   * Explore the data models and flow of JSON requests and responses.
2. **Implement Prompt Formatting**: Write prompts for generating MCQs based on a topic and complexity level.
3. **Integrate OpenAI API**: Complete the OpenAI API integration to dynamically generate MCQs and evaluate user submissions.
4. **Test and Debug**: Validate the functionality of each endpoint, ensuring correct processing of JSON data.

**Task Solution**

Update boilerplate **mcq.py** code with two endpoints:

**Challenge 1: (**/mcq/generate)

1. Write the appropriate prompt to generate multiple choice questions for the selected topic and selected complexity
2. Call OpenAI API to execute the prompt.

**Challenge Solution 1.a:**

**Replace**

prompt = f"{request.topic} {request.complexity}"

**with**

prompt = f"""Create a multiple-choice interview question about {topic} with complexity level: {complexity}.

    Return the response in the following JSON format:

    {

        "Id": "string (e.g., Q1, Q2)",

        "Question": "string containing the complete question text",

        "Options": [

            {

                "OptionIndex": 0,

                "OptionValue": "string containing first option"

            },

            {

                "OptionIndex": 1,

                "OptionValue": "string containing second option"

            },

            {

                "OptionIndex": 2,

                "OptionValue": "string containing third option"

            },

            {

                "OptionIndex": 3,

                "OptionValue": "string containing fourth option"

            }

        ],

        "CorrectOptionIndex": "number between 0-3",

        "Complexity": "string (Basic, Intermediate, or Advanced)"

    }

    """

**Challenge Solution 1.b:**

**Replace**

 response = {}

**with**

response = openai.ChatCompletion.create(

model=" gpt-4o-2024-05-13",

temperature=0.5,

messages=[

{"role": "system", "content": "You are a technical interviewer creating MCQs..."},

{"role": "user", "content": prompt}

]

)

# Extract and process the generated text

result\_text = response.choices[0].message.content.strip()

**Challenge 2(/mcq/submit)**

a) Generate the prompt to submit multiple choice questions and get feedback along with rating between 1 and 10

b) Call OpenAI API to get the response.

**Challenge Solution 2.a**

**Replace**

 prompt = f"""

    {candidate\_answers}

    """

**with**

prompt = f"""

Evaluate the following candidate's answers to multiple-choice questions and provide:

- An overall rating out of 10.

- Specific feedback on each answer, highlighting strengths and weaknesses.

JSON Data:

{candidate\_answers}

"""

**Challenge Solution 2.b**

# Evaluation API Call

feedback\_response = openai.ChatCompletion.create(

model="gpt-4",

temperature=0.2,

messages=[

{"role": "system", "content": "You are an expert in evaluating multiple-choice questions. Provide detailed feedback on each answer."},

{"role": "user", "content": prompt}

]

)

By following these steps, students will enhance the functionality of the mcq.py script, enabling dynamic MCQ generation and robust evaluation.