Quiz Game

1. Introduction

This is a quiz game is interactive single player game. This is an online game where player needs to login to the site to start playing. The player login required here as the game supports playing at different times, user can answer few questions save and come back later to continue. In order to provide this user experience, we need to have an account and save user game status. The game presents player with multiple choice questions and they enter answer by entering an option number.For every right answer score increases and incorrect has score reduced. At the end of the game user will be able see how many questions were right and how many were not , and also the total score for the game.

2. Design Implementation

Quiz Data

First thing is to understand how to represent the quiz data, how to get test data --> read from a file or read from end point but format of this data should be defined

sample dataset =

```
{
  "quiz_data":[
      {
       “Id” : 1,
      "question":"Which is the capital of California?",
      "options":[
        "san diego",
        "los Angeles",
        "sacremento",
        "san francisco"
      ],
      "answer":2
    },
    {
      "id":2,
      "question":"Which is the capital of Oregon ?",
      "options":[
        "portland",
        "bend",
        "salem",
        "beaverton"
      ],
      "answer":2
```

```
    }
  ]
}
```

1. question ----> string
2. Options —> a list with 4 options of answer to choose from / to present to user
3. Correct answer —> int (index at which the answer is in the options)

Quiz building model /objects
Build a question model class and build question data set/object

Quiz class

Initialize
1. score = 0
2. num_of_questions = 0
3. current_question = ""
4. options = []
5. questions_list =

  Methods

1. still_has_question() → checks if questions are left, continues to present questions
2. get_next_question() → gets next question and options to present
3. check_if_answer_true() ---> Update_score

UI Class

print(question and option to users)
print(remaining_question and current total score)

User class
Authentication

main.py
1. build question data

2. while still_has_question()
        get_next_question()
        print/show(question and options)
        get_user_answer()

check_if_answer_correct()
show_message_to_relfect()

3. Done with all questions, finish the game

Enhancements:
save and exit, when the user comes back continues with unanswered question and continues with remaining questions.