Documentation design description:

Assignment 5

 Email from all guys in group sign off confirmation:

|  |
| --- |
| **Coffey, Alexandria Marie** <amcoffey@mun.ca>  1:57 PM (3 hours ago)  to    Hey guys just checking to see if my part looks good and has been working good With your code?!  2    **Asaolu, Dayo David**  2:42 PM (2 hours ago)    to    Yes, all good      **Chris Smith**  2:55 PM (2 hours ago)  to    Looks good to me. Chris |

Programmer part 2: Create Quiz

For this assignment, I have been tasked with the responsibility to create the quiz, with the instructions that the instructor only have the authority to modify any details.

To begin, I created two classes, a question class and a Quiz class both of which contain some arguments holding the responsibility on what the task of each class is to do.

The quiz class has attributes *quizID (to return the user ID of the student), Current*\_AllowedAttempts (to return the number of times the quiz can be attempted), and \_allowedTime (which returns the value of how long the quiz can be open/accessed for).

For each argument, I created more functionality that allows the instructor to modify any existing information that is needed to create the quiz.

The question class is the system user class that is being used by all programmer parts for this assignment. The methods of the question class are below and explained what each one will do:

*Show\_questionText - return text of question  
show\_PossibleAnswers - return list of possible correct answers  
getCorrectAnswers - return list of correct answers  
modifyValue - return value of question  
setQuestionText - set question text  
setCorrectAnswers - set list of correct answers*

My part is to make sure that the quiz is eligible for the instructor only when they plan on making a quiz from scratch. They should be able to just access questions, answers, possible answers, correct answers all from a question bank and modify them each time. Each method that is provided, is simplified and decoupled in simplest way that allows the user to automatically (programmer part 4) do this.

Create the quiz, enter questions and values and deliver them to students. The whole project connects together, login and authenticate is the main piece that gets everything connected together and all the other pieces working.

There are multiple tasks that this part require, such as modifying quiz start/end time, being able to create new questions, creating the quiz itself (like I done with a while loop in the MakeQuiz) function.

USE CASE for programmer part 2 attached.

Requirements:

I will include all the requirements for the whole team project because they all connect together and depend upon one another, but I will elaborate on my programming part, part2 and the required functions needed to specifically create the quiz.

**Requirements List**

1. **Functional**
   1. **User Authentication**
      * **Users can authenticate with Login/Password**
      * **Two levels of access: instructor and student**
      * **New accounts can be configured**
      * **Users can authenticate with their MUN credentials.**
      * **User can authenticate with either MUN/SuD credentials and will be granted access accordingly**
      * **Users can request a change in authentication Password.**
   2. **Instructors can create a quiz: This is the programmer part two that I have, so for this part the instructor is the primary actor (The only user)** 
      * **Instructors can create a quiz.**
      * **Quiz attributes**
        1. **Dates**
        2. **Attempts**
        3. **Multiple Answers**
        4. **Weighting**
        5. **Student list**
      * **Questions are entered in Quiz Bank**
      * **Quiz creators are able to have multiple versions of a question**
      * **Quizzes creation can be done with different types of questions**
      * **Questions may be given in a random order.**
      * **Instructor can add dates.**
      * **Quizzes can be assigned time limits.**
      * **Instructor can hide or display results of the quiz.**
      * **Instructor can specify number of possible quiz attempts.**
      * **Instructor can extend time limit of the quiz.**
      * **Instructor can modify the existing quiz.**
      * **Quizzes should notify time remaining.**
      * **User with editing permissions can add images to quiz questions**
      * **User can remove images from quiz questions**

**This is the requirements for creating the quiz: Each of which are done in functions in the code. The dates, attempts, multiple answers, multiple versions of a question, different question text, time limits, possible attempts, modifying the existing quiz, time remaining…etc All are elaborated on in my docstrings and functions there that allows the instructor or automatic operator to do so for creating the quiz.**

* 1. **Users can take a quiz.**
     + **Users can take permitted quiz**
     + **Access limitations are checked**
     + **Users can search for a quiz by its name**
     + **Users can have multiple attempts.**
     + **Users can see number of completed attempts of a certain quiz.**
     + **Users can tick the question(s) that they don't understand.**
     + **User leaving the quiz page will be recorded.**
     + **Users can select other quiz questions to complete.**
     + **User can get a hint on a question**
     + **User can retake a quiz (For no credit)**
     + **Users can select total number of quizzes they must answer.**
     + **User can save their answers.**
     + **Users can see quiz time remaining**
     + **User can pause quiz.**
     + **Instructor can can provide a bonus question for select students.**
     + **Users can listen to quiz questions using a Text-to-Speech feature**
     + **Quiz Submission**
       1. **All answers are auto-saved upon quiz submission**
       2. **A confirmation notication for submission appears**
       3. **User gets a warning when trying to submit not finished quiz.**
       4. **User's answers to each question is auto saved and submitted.**
  2. **Quiz Results**
     + **Student View**
       1. **Students can view solutions to completed quiz**
       2. **Students can view class average for a quiz.**
       3. **Student can email the professor after a quiz.**
       4. **Users can check the grades of their submissions**
       5. **View attempts**
       6. **students can rate difficulty of completed quizzes**
     + **Instructor View**
       1. **Instructor views class result statistics**
       2. **Instructor views attempt statistics**
       3. **Instructor views student statistics**
       4. **Instructor can view question grading statistics.**
       5. **Instructors can sort student list by quiz grade**
       6. **Instructors can give feedback to students**
       7. **Instructor can see the average time it took for students to complete each quiz**
     + **Marks and Grading**
       1. **Students must have 50% grade to pass.**
       2. **Highest score will be recorded as a final grade**
       3. **Student user can see their average amongst all quizzes**
     + **A log entry for quiz result will be generated by SuD.**
     + **Student's can request excused absence**
     + **Tech wants all grade changes to be tracked**

Here is the description of our whole project and how it will all work together and will use the DRY concept with no decoupling and have high cohesion.

All you basically need to know to use this program is that this project is based on an online quiz, that lets a user LOGIC/authenticate which is (programmer part 1), it lets the instructor only create the quiz which is (programmer #2), also lets the students/users take the quiz(Programmer #3), and lets them review results(student view) and lets the instructor post class results, class attempts and individual results(programmer #4). So here is my basic description of my module:

To begin this project you need a flask based application downloaded on your operating system and just basic python 2.9 or greater. You will not need any packages or files downloaded, all you need is a system to run python code and flask on. Programmer part 2 is responsible for creating the quiz with a function (def makeQuiz) that uses a while loop to create the quiz that allowes the instructor **ONLY** to create or modify quizzes.

Dealing with the persistence interface, the create a quiz class needs to have a method to get the user ID from the login/authenticate/persistence programmer part 1. If the userID with username and password is a Instructor, return Boolean TRUE, else return FALSE. The method getUserID is needed from the persistence interface to check if the userID is an instructor only, so that only an instructor can access/modify/change any quizzes.

Programmer part two basically underlines the whole project, the methods defined here are used throughout each part because the other programmers need to have functions for questions that allow them to create and modiy other users, questions, start times all for the quiz.

For this part I need to create MCQ questions and answers and have them stored in a question bank so that they can be accessed differently and as frequently as needed, every time. Each time a question is stored in the question bank it means that we can just grab any random, modified questions and use them. So basically, what I have done here was created a list of questions under quiz class, and made a function to create the quiz that lets you enter the question and correct answers and will allow you to access these from the quiz bank function. Hope this helps!

The deliverables are provided in doc strings under each class functions.