Clifford Bowe

COMP 2005 – Assignment 6

***CHANGES: The base implementation of the module and its intended effects haven’t changed, thus the below information is mostly unchanged as it is still valid.***

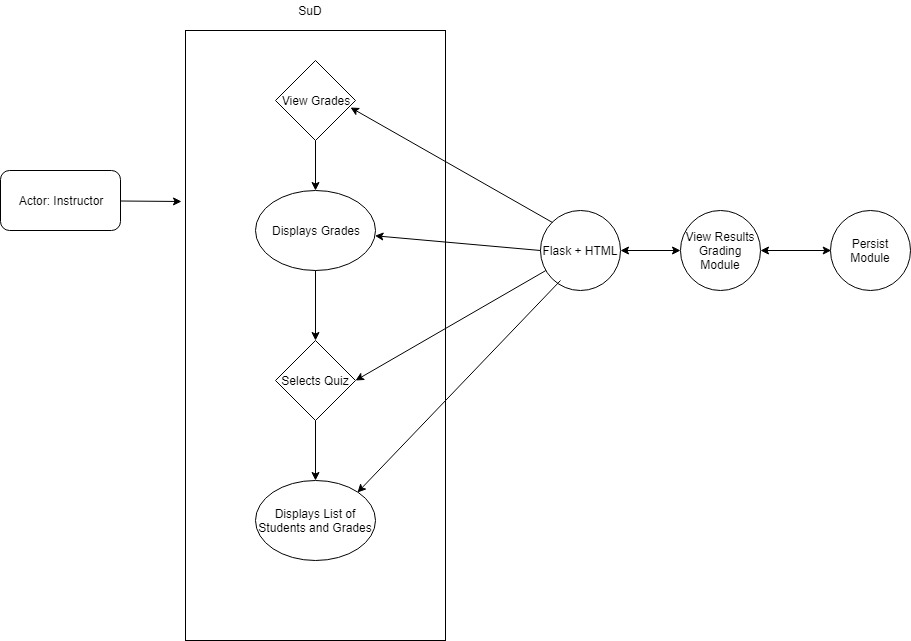
***The QuizAttempt module and a sample QA output dictionary were used for demonstrative purposes, due to this being an individual submission. The lines of code that interact with Persist are denoted by a triple pound sign (###) in the module itself, while the sample lines are denoted with comments describing why they’re there.***

***The original Assignment 5 deliverables were submitted as a zip file in the directory for reference.***

**Description of Programming Component**

I am handling the View Results/Automatic Grading portion of the design for the Quiz application. My component will automatically grade the quiz submitted by the student, while also allowing the instructor to view results in several ways (class participation, average, histogram) and also allowing the students to view results in several ways (summary by quiz, summary by quiz of all attempts). It will do this by storing the quiz attempts in a dictionary with the quiz name as the keys for the dictionary. Each key will access a list that holds the individual attempts for each quiz. Each function will then perform the necessary calculations to deliver the desired output.

**Use Case UML Diagram**



**Clear Connection between Module and Functionality**

The basis of my implementation is going to be the “\_quiz\_dict” dictionary member variable of my QuizResult object. When called, it will take the dictionary from Persist, supplied by the Take Quiz module, and iterate over it, adding each attempt’s information to the \_quiz\_dict dictionary, with the quiz names as keys and a list with the attempt data (student email, list of questions, list of choices, list of correct answers and list of student responses) for each individual attempt

Per the requirements list (<http://www.cs.mun.ca/~brown/cs2005/Notes/project/projrequirements.html>) each of my functions is tailored to satisfy a specific requirement, which should be clear from the naming of the function. For example, the ins\_class\_part function is meant to return the class participation as a percentage value to the instructor. The stu\_quiz\_grades function is meant to return a list of strings representing a summary of the student’s grades for a specific quiz. The naming convention makes the connection clear. The flask front end will then use these functions to display the appropriate data as needed.

**Description of Module Design**

The module is designed such that the information detailed in the mandatory requirements list is all linked to an individual function. For example, for the instructor to view the class average of a quiz, the ins\_class\_avg function exists and will be called to return the data for the histogram as a list. It will then be displayed through the flask and HTML based front end as required.

For the purposes of implementation, ToD would simply need to verify that the dictionary being received from the Persist module is of the proper format below:

Take\_Quiz\_Export\_Dict

Keys: Student Emails

Values: Dictionaries

Keys: Quiz Names

Values: Quiz Attempt Objects

The quiz attempt object structure is laid out in the Take Quiz documentation.

The flask front end, once fully constructed, will simply call the functions as needed to display the required data for the given web pages.