CMPT 561 Project – Fall 215

Phase 1: Design a Rubric-based Evaluation System (eRubric)

Please post questions to Piazza about any ambiguities in this document then I will add further clarifications and post an updated document. **Note that the project is worth 30% of the overall grade** (10% for each phase). **Push your work to GitHub as you make progress.**

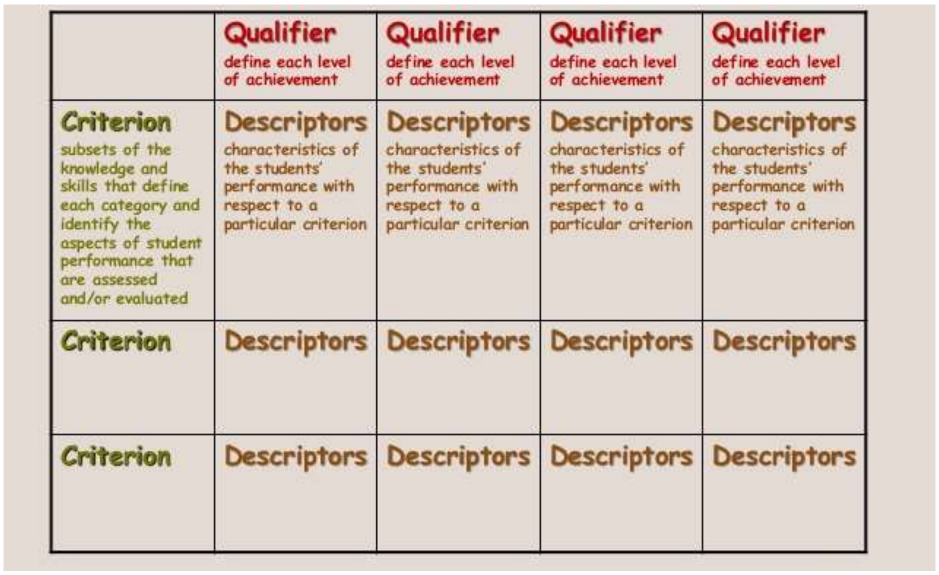
Project Phase 1 due date is by 5pm **Thursday 29h October**.

1. Introduction

In the academic context, rubrics help instructors evaluate student work consistently and objectively by listing evaluation criteria for any given assignment. They help to clearly communicate expectations of quality for an assigned task. Rubrics can help students organize their efforts to meet the requirements of an assignment, while faculty can use rubrics to explain their evaluation to students. During grading, as scores and feedback are entered into a Rubric the overall score can be automatically computed, which becomes a considerable time saver.

Rubrics are useful in two ways. First, effective rubrics require the instructor to enumerate exactly what are the qualities and criteria that the student is expected to achieve in each grading category. The second way rubrics are helpful is to provide a grading framework with richer feedback to the student. This fosters accurate and fair assessment. Students can even user them for self-evaluation and reflection before submission!

As shown if figure 1, a scoring rubrics include one or more dimensions on which performance is rated, and a rating scale for each dimension. Dimensions are generally referred to as **criteria**, the rating scale as **levels**, and definitions as **descriptors**.



	Qualifier define each level of achievement	Qualifier define each level of achievement	Qualifier define each level of achievement	Qualifier define each level of achievement
Criterion subsets of the knowledge and skills that define each category and identify the aspects of student performance that are assessed and/or evaluated	Descriptors characteristics of the students' performance with respect to a particular criterion	Descriptors characteristics of the students' performance with respect to a particular criterion	Descriptors characteristics of the students' performance with respect to a particular criterion	Descriptors characteristics of the students' performance with respect to a particular criterion
Criterion	Descriptors	Descriptors	Descriptors	Descriptors
Criterion	Descriptors	Descriptors	Descriptors	Descriptors

Figure 1. Anatomy of a Rubric

2. Requirements

Design and build a system to share Evaluation Rubrics and allow using them to evaluate a list of Work items (e.g., evaluate a project implementation submitted by students). This system should focus on the use cases shown in Figure 1 and explained in Table 1. The aim is to facilitate the use of good techniques in computer assisted assessment.

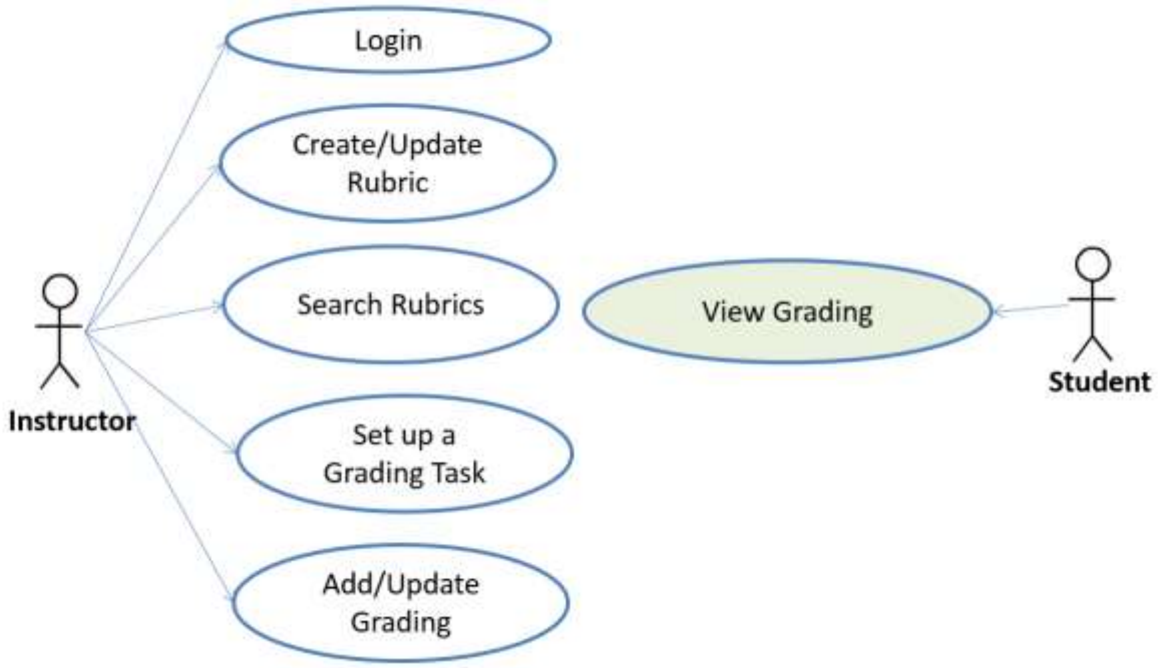


Figure1. eRubric use cases

Note some use cases are omitted or simplified to keep the project scope reasonable. Also, assume that the system already has the list of instructors who can use the system. Hence, no need to provide the ability to maintain these entities unless explicitly stated in the requirements.

Table 1. eRubric use cases description

Use Case	Description
Login	The instructor should first Login before using eRubric. To login, the user should enter their username and their password. The system should validate the entered credentials. If the login fails then an error message should be displayed and the user should be prompted to login again. If the login is successful, the system should display the menu. The users should only be granted add/update grading).
Create/Update Rubric	Allow the user to create or update a rubric using an intuitive Rubric Editor. A rubric should have: • Rubric title and description • Enter a short title briefly describing the rubric in Rubric Title and a longer description in Description. • Keywords associated with the rubric. • Rubric primary subject (e.g. Computer Science) and category (e.g. Design document). • Rubric levels and their weights. A simple weight scale for levels would be 1, 2, 3, 4, etc. with 1 being the poorest performance level.

	• Rubric Criteria and their weights. Weights can be a percentage assigned to distribute the scoring among criteria. The total weights should be 100%.
Search Rubrics	Get the list of all available rubrics or filtered by category. The user can select a particular rubric to display its details.
Set up a Grading Task	1) Upload the list of students to evaluate. The list of should have: • StudentId • First name • Last name • Email • GroupId (could be empty for individual work item) • Password 2) Enter Course Code and Course Name 3) Enter a Title and Description of the submitted work item to be evaluated 4) Select Individual of Group Evaluation 5) Select the Evaluation Rubric to be used
Add/Update Grading	Use the evaluation rubric to evaluate the work item for each student/group by entering/updating a score for each rubric criteria. Store the grading details and compute the overall score.
View Grading	Allow a user to use a direct Web link to view the grading results both the details and the computed summary.

3. Deliverables

- Identify and design the forms required for your Web interface and the flow between the forms. **Discuss your design with the instructor before the implementation.**
- Design and develop your model and document it using a class diagram.
- Design the Entity Repository methods and Design eRubric controllers to meet the project requirements.
- **Design eRubric Web UI:** the required HTML forms, navigation,
- Write a design and testing document to include the class diagram and screen shots of conducted tests.
- Demo your implementation and answer questions about the implementation. 20 minutes demo will be allocated to each team.

4. Grading

Your project will be graded based on the **completeness** and the **quality of the implementation**. In order to receive full credit in each area, it must be **1) complete, 2) done well, and 3) tested**. Below is the breakdown of the grading criteria and it will be further refined.

Grading Rubrics

Criteria	%	Functionality*	Quality of the implementation

Complete, correct, accurate and good quality implementation of the model	7		
Complete, correct and accurate implementation of Entity Repository methods	7		
Complete, correct and working implementation eRubric use cases - Login and Home page with Menu	6		
- Create/Update Rubric	10		
- Search Rubrics	24		
- Set up a Grading Task	10		
- Add/Update Grading	16		
- View Grading	10		
Design documentation Class Diagram showing **Entities**, **Repositories** and **Controllers**	5		
Testing documentation with evidence of correct execution using snapshots illustrating the results of testing to show that your implementation works and meets the requirements.	5		
Total	100		
No demo of the implementation	-50%		
Not submitting the design and testing documentation	-30%		
Not using the design and testing template	-10%		
Copying and/or plagiarism or not being able to explain or answer questions about the implementation	-100%		

* **Possible grading for functionality**: ***Working*** (get 70% of the assigned grade), ***Not working*** (lose 40% of assigned grade and ***Not done*** (get 0). The remaining grade is assigned to the quality of the implementation. In case your implementation is not working then 40% of the grade will be lost and the remaining 60% will be determined based on of the code quality and how close your solution to the working implementation. Code quality includes **correct usage of MVC**, applying OOP best practices particularly encapsulation, inheritance and polymorphism when relevant, meaningful naming of identifiers, no redundant code, simple and efficient implementation, clean code without unnecessary files/code, use of comments where necessary, proper white space and indentation.

Marks will be reduced for code duplication, poor/inefficient coding practices, poor naming of identifiers and unnecessary complex/poor user interface design.

5. Submission Guidelines

- Your design and testing Word Document **must follow the provided template**. It must be placed in a ***docs*** folder within your implementation project.
- Your implementation should be pushed to GitHub as you progress. Every team member should actively contribute to the project. I will assess each team member contribution based on the files they push to github. Potential free riders will be easy to spot.

- **You must submit a hardcopy of your design and testing document during the demo session. I will grade using the hardcopy.**

Important Notes:

- It is very critical to pay attention to the proper application of MVC.
- All assignments **must be your own original work.**
- Each team must schedule at least one office hour meeting with the instructor to review and discuss your implementation and get feedback before your submit your work.
- For any email you send me w.r.t. the project please CC all the team members also add CMPS356 to the email title.
- **No free ride is allowed!** All students must contribute to the best ability to the success of the projects. Team work skills are critical. Please help each other, learn from each other and keep a good team spirit. If the team complains about a student's poor contribution then he/she will be asked to submit his/her own solution individually.
- **All team members need to participate and be present during the demo of your solution.**
- **Office hours are your right**. Please use them and come and see me if you need any further clarifications and guidance (not solutions!).
- **Late submissions** will result in **severe point penalties**. If you submit one day late (or less), 10 points will be deducted from your grade, 2 days will deduct 30 points, and any submissions after that will receive an automatic zero.