CMPT 561 Project – Fall 215

Phase 2: Implementation and Testing of eRubric (Rubric-based Evaluation System)

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 2 due date is by 11pm **Saturday 28th November**.

1. Phase 2 Requirements

Implement and test eRubric use cases following a Single Page Application (SPA) design and using Angular 2, JavaScript (ES2015) and TypeScript. You need to create a modular solution using TypeScript, Angular 2 and ES2015 features particularly modules, classes, promises, arrow functions, etc. For the server-side components you can either use ASP.Net MVC 6 or Nodejs. For the data store you may use either a relational database or MangoDB.
The implementation should deliver 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.

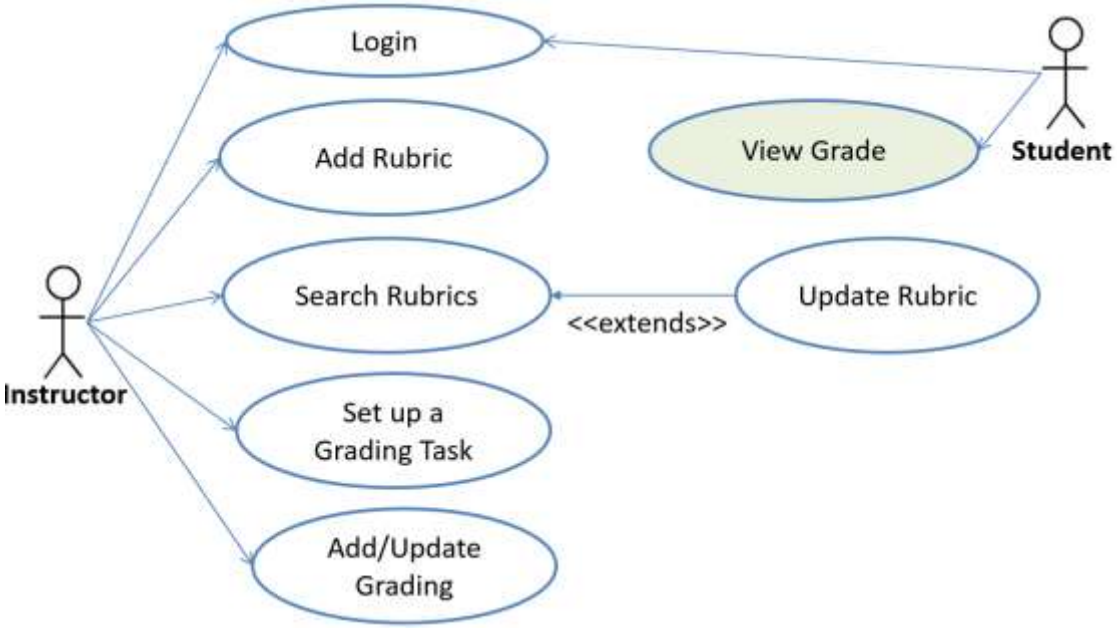

Figure1. eRubric use cases

Note some use cases are omitted or simplified to keep the project scope reasonable. Also, assume that the data about instructors, courses, students, Instructor Courses and Student Courses are provided via BannerService. Hence, no need to provide the ability to maintain these entities but you need to design the service interface of BannerService to get the required data for eRubric.

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 the use cases they have access to.

Add/Update Rubric	Allow the user to add or update a rubric using an intuitive Rubric Editor.
	A rubric should have:
	• Rubric title and description
	• Keywords associated with the rubric
	• Rubric primary subject (e.g. Computer Science) and category (e.g. Design document).
	• Rubric levels, their coefficient range and default coefficient. For example:
	  o Excellent (0.9 to 1) – default 1
	  o Good (0.8 to 0.89) – default 0.8
	  o Satisfactory (0.6 to 0.79) – default 0.6
	  o Poor (0 to 0.59) – default 0.5
	• 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 or update its details.
Set up a Grading Task	1) Select the Course to do the grading for 2) Enter a Title and Description of the assignment to be graded 3) Select Individual of Group Evaluation. In case of group evaluation the instructor needs to setup the student groups. 4) Select the evaluation Rubric to be used
Add/Update Grading	• Get the list of students or groups to be graded. • Use the evaluation rubric to evaluate the assignment for each student/group by selecting/updating a level and a coefficient for each criterion of the rubric. If grades were previously entered for the selected then the system should allow the instructor to change them otherwise an empty grading sheet should be provided. • Enter optional overall comment for each graded assignment • Store the grading details and compute the overall score. • Navigate backward and forward through the list of students to add/update their grading or the instructor can select a particular student to add/update their grading (without leaving the grading view).
View Grading	Allow the student to view the grade details and the computed summary for a particular course and assignment.

2. Deliverables

- Document in details 5 lessons learned by comparing your submitted implemented with the model solution provided. You need to provide detailed reflections about the new concepts and skills acquired from phase 1 of the project.
- Identify the required REST services and implement them (besides implementing AuthenticationService, BannerService). The interaction between the client components and the backend should be done using REST services.
- Implementation and testing of eRubric use cases.
- Manage eRubric data in a relational or a MongoDB database.
- Draw the system's overall architecture, the class diagram of the REST Services and eRurbric Database ER diagram/schema.
- Write a testing document including and screenshots of conducted tests illustrating a working implementation.
- Demo your implementation and answer questions about the implementation. 20 minutes demo will be allocated to each team.

3. 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 and accurate design of AuthenticationService, BannerService and other required services.	8		
Complete, correct and working implementation eRubric use cases - Login and Home page with Menu	6		
- Add/Update Rubric	15		
- Search Rubrics	10		
- Set up a Grading Task	10		
- Add/Update Grading	25		
- View Grading	10		
5 lessons learned from Project Phase 1	5		
Design documentation (1) System's overall architecture (2) REST Services Class Diagram (3) eRubric Database ER diagram/Schema	6		

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. Design quality includes **correct usage of SPA**, applying OOP best practices particularly encapsulation, inheritance and polymorphism when relevant, meaningful naming of identifiers, no redundant code, simple and efficient design, 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.

4. 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:

- 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 CMPT 561 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.