Web App

CMPS356 Project Phase 1 – WebApp Design

This is a group project worth 10%. The project submission is due by 5pm Thursday 6th April 2017.

1. Requirements

Students have lot on their plate! What assignment is due? Is there any in class assessment today? When is my midterm? Sometimes it is hard to keep up with everything and remember everything you have to do. Moreover, some instructors tend to forget that students have other classes to study for and have a life to enjoy! On the other hand, some students tend to leave things to the last minute!

As part of the university recent _Road to Student Success_ initiative, you are requested to design and implement a workload management web app named **RIFQ** to record and manage the graded tasks associated with each course. The aim to ensure that students are well informed about the workload associated with their courses so that they can better manage their time. Additionally, RAFIQ will help avoid overloading students with too many deliverables particularly the ones that are due at the same time. RIFQ is intended to boost students' performance and success, improve progression, and raise graduation rates. RIFQ requires instructors to enter the graded tasks (e.g., quizzes, assignments, projects, exams) associated with the courses they teach during a semester. Then the Program Coordinator and/or students can post comments to report issues or to request workload changes for the instructor's consideration.

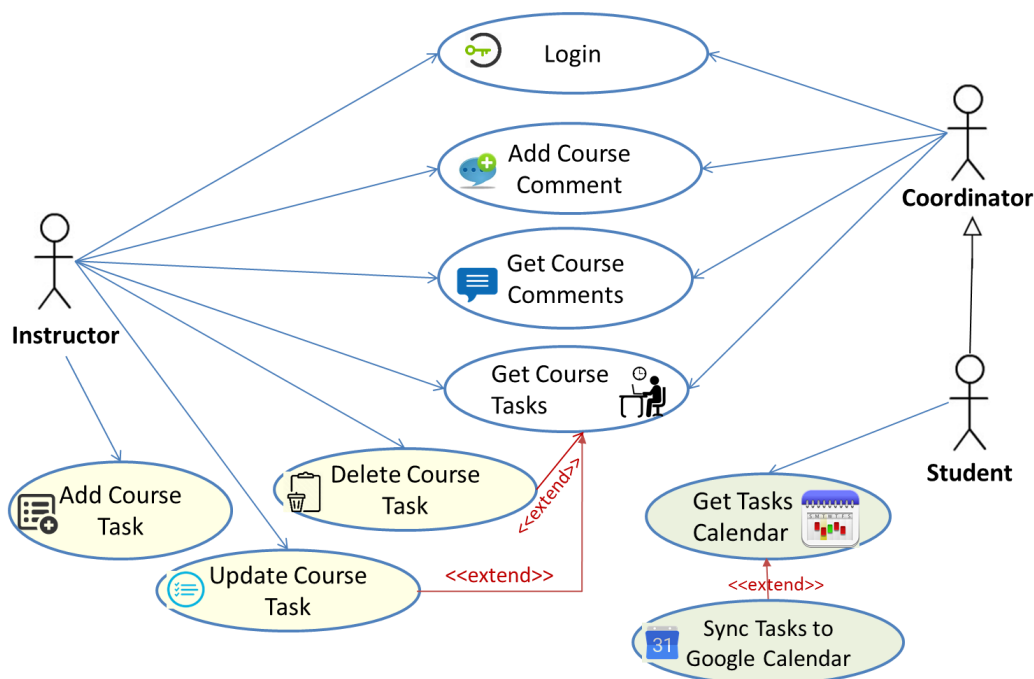The key use cases to deliver are shown in Figure 1.

Figure 1. RIFQ Use cases

Table 1. Use cases description

⚷ **Login**	Allows the user (i.e., *Coordinator*, *Instructor* and *Student*) to login to use the application.
Add Course Task	Instructor can add graded tasks (e.g., quizzes, assignments, projects, exams) associated with the courses they teach during a semester. When adding a task, the instructor should be able to select the course and assignment type (i.e., Homework, Quiz, Midterm exam, Final exam, Project). For each task, the instructor should enter the Due date, the Average Number of hours required to complete the task and Percentage towards Final Grade. - When entering homework, the system should ask for the number of homework assignments and the due date for each one. - When entering project, the system should ask for the number of project phases and the due date for each phase. Also, the user should specify the Student Group Size. - When entering quiz, the system should ask for the number of quizzes and the date for each one. - When entering midterm, the system should ask for the number of midterms and the date of each one. Validation rules: - Only final exam can be added - At most 2 midterm exams can be added - No two tasks for a course can have the same due date - Project phases should be between 1 to 4 - Maximum 8 homework assignment per course
Update Course Task	Update a task associated with a course. Update should use the same UI as Add Task.
🗑 **Delete Course Task**	Add a task associated with a course.
Get Course Tasks	Get the tasks associated with a course of set of courses: - The instructor can only access the tasks of the courses they teach for a particular semester. They can get all the tasks or filter them by course. - The coordinator can access the task of any course or the tasks associated with courses belonging to a particular program (e.g., task for CS courses). - The student can only access the tasks associated with their courses. They can get all the tasks or filter them by course.
Get Tasks Calendar	Get tasks calendar display the due tasks organized into 4 classifications: due today/tomorrow, due next week, due in next 2 weeks and others.

|	
31 **Sync Tasks to Google Calendar**	Student sync their course tasks timetable to their Google Calendar to be reminded of important due dates.
Add Course Comment	Program Coordinator and/or students can post comments to report issues or to request workload changes for the instructor's consideration. The instructor can also add course comments to respond to feedback and provide justifications. The comment should have a title and a body. For simplicity, comments are associated with a course and not a task.
Get Course Comments	Users can get comments associated with a course.

Deliverables:

1) Architecture Design, Classes Design and the UI design to deliver RIFQ use cases.
   The design documentation should include at least the following:
   • Application Architecture Diagram
   • Class Diagram showing Entities, Repositories and Services and Controllers.
   • UI Design and navigation.
   • Discussion of design rationale (i.e., justification) of key design decisions.
   During the weekly project meetings with the instructor, you are required to present and discuss your design with the instructor and get feedback. You should only start the implementation after addressing the feedback received about your design.

2) Implement the client-side to deliver RIFQ use cases based on your previously developed and validated design. RIFQ web pages should use HTML 5, CCS and JavaScript without any server-side interaction (the latter will be done in phase 2). The pages should comply with Web user interface design best practices. Also, remember that 'there is elegance in simplicity'.

Push your implementation and documentation to your group GitHub repository as you make progress.

2. Grading rubric

Criteria	%	Functionality*	Quality of the implementation
Application Design Architecture Design, Classes Design and the UI Design to deliver RIFQ use cases. The design documentation should include at least the following: • Application Architecture Diagram	20		

• Class Diagram showing Entities, Repositories and Services and Controllers. • UI Design and navigation. • Discussion of design rationale of key design decisions.			
Complete and correct implementation of the requirements:			
Add Course Task	22		
Update Course Task	16		
Delete Course Task	2		
Get Course Tasks	12		
Get Tasks Calendar	10		
Sync Tasks to Google Calendar	0		
Add Course Comment	6		
Get Course Comments	6		
Testing documentation with evidence of correct implementation using snapshots illustrating the results of testing.	6		
Total	100		
Copying and/or plagiarism or not being able to explain or answer questions about the implementation	-100%		

* **Possible grading for functionality**: *Complete and Working* (get 70% of the assigned grade), *Complete and 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. Quality includes **correct application of MVC**, 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.

3. Ground Rules

- All assignments **must be your own original work**, not based on the work of other students, online examples/tutorials, or any other material from any other source. Any assignments found to be based on work other than your own will automatically be given a **grade of zero**, and may lead to further disciplinary action as per QU policy.

- All assignments must be submitted electronically to Github. You should push your work to Github as you make progress. Late submission policy: 10 points deduction for each late day and 0 after 3 days.

Appendix

Resources:

- Courses list https://cmps356s17.github.io/project-data/courses.json
- Students list https://cmps356s17.github.io/project-data/students.json
- Instructors list https://cmps356s17.github.io/project-data/instructors.json
- Sample Course Tasks https://cmps356s17.github.io/project-data/tasks.json