Web App

CMPS356 Project Phase 1 – WebApp Implementation

This is a group project worth 10%. The project submission is due by 8am Sunday 23th April 2017.

You are required to implement and test RIFQ Web App that you have designed in phase 1 (refer to project phase 1 document for further details). You will deliver the same use cases as phase 1 by extending the provided base solution.

Deliverables:

1) Document in details 5 lessons learned by comparing your submitted project phase 1 with the model solution provided. You need to provide detailed reflections about the new concepts and lessons learn when you compare your submission with the model solution.
2) Implement the client-side and the server-side Web components to deliver RAFIQ use cases based on your previously developed and validated design.

RIFQ should be fully implemented using Node.js. The application data ~~can~~ should be managed using the provided *json* files:

- Courses list https://cmps356s17.github.io/project-data/courses.json (one course has sample tasks and comments)
- Students list https://cmps356s17.github.io/project-data/students.json
- Instructors list https://cmps356s17.github.io/project-data/instructors.json
- Tasks types https://cmps356s17.github.io/project-data/taskTypes.json. Task Type dropdown should be dynamically loaded from the taskType file. When adding/updating a task, validate the maximum allowed tasks per type using the max property associated with each taskType.

The app implementation should follow MVC pattern. Also, remember that 'there is elegance in simplicity'.

3) Test and document the testing of your solution.

Important notes:

- When adding a Comment you should auto-set the:
  - CreatedBy to current logged-in user
  - CreatedDate to current date and time
- The menu should appear in every page except the login page. The menu and needed styles and common JavaScript should be shared between pages and placed in a layout template.
- Same form should be used for add/update task.
- Task type should be a dropdown and it should be filled with data from the file @ https://cmps356s17.github.io/project-data/taskTypes.json Note that this file also include the maximum number of allowed tasks per task type (e.g., maximum 10 quizzes per course)
- Do not create **entity** classes (just use objects without creating classes). Only need to create Repository and Controller classes.

- To keep it simple you may organise your implementation in 2 repositories (and corresponding 2 Controllers). The minimum methods of each repo could be as shown in the Diagram below.
- Continue posting your questions https://piazza.com/qu.edu.qa/spring2017/cmps356/
- Push your implementation and documentation to your group GitHub repository as you make progress.

1. Grading rubric

Criteria	%	Functionality*	Quality of the implementation	Score
Complete and correct implementation of the requirements:				
Login and Customized Navigation Bar	8			
Get Tasks	10			
Add Task	12			
Update Task	8			
Delete Task	6			
Student Tasks Calendar	10			
Sync Tasks to Google Calendar	10			
Get Comments	8			
Add Comment	8			
Workload Summary Report	10			
Documentation - 5 lessons learned from Phase 1	5			
Testing documentation with evidence of correct implementation using snapshots illustrating the results of testing (you must use the provided template).	5			

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.

2. 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.