



## Web App

### CMPS356 Project Phase 3 – WebApp Implementation using Angular and MongoDB

**This is a group project worth 10%. The project submission is due by 8am Sunday 28<sup>th</sup> May 2017.**

You are required to re-implement and test RIFQ Web App that you have designed in phase 1 and implemented in phase 2 (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. The refactored implementation should follow SPA architecture and it should use Angular for client-side UI and MongoDB for managing the data.











#### Deliverables:

- 1) Compare and reflect on differences between the MVC-based design and implementation submitted in phase 2 with SPA-based design and implementation submitted in this phase. Document and discuss in details 4 major differences. You need to provide detailed reflections about the comparison of the two models.
- 2) Provide SPA architecture diagram for your overall design.
- 3) Design ~~and draw~~ the Database schema then implement the Database schema to manage the data in a MongoDB database. The implementation should use mongoose and Node.js.
- 4) Populate the database with the data from the json files.
- 5) Implement the Repositories to offer the same functionality as phase 1 but it should use MongoDB as the data source. **All data filtering should be done by MongoDB server and only the required data should be retrieved.** The implementation should make use of MongoDB capabilities (e.g., ~~ability to populate referenced entities using the find(...).populate(...)~~ using aggregate query to get the data for the aggregate report).
- 6) Implement the client-side of RAFIQ using Angular. Make the necessary changes to the server-side implementation to provide the needed services.
- 7) You need to test as you go and gradually change the Controllers to use the new Repositories.
- 8) Write unit tests to test your server-side repositories, ~~server-side and client-side services~~.
- 9) Write a testing document including and screenshots of conducted tests illustrating a working implementation.
- 10) Demo your implementation and answer questions about the implementation upon request.

#### Important notes:

- Continue posting your questions <https://piazza.com/qu.edu.qa/spring2017/cmeps356/>
- Do not forget to submit your design and testing documentation (in Word format) and fill-up the Functionality column of the grading sheet using the provided phase 3 template.
- Push your implementation and documentation to your group GitHub repository as you make progress.

## 1. Grading rubric

Criteria	%	Functionality*	Quality of the implementation	Score
<b>Database Implementation and Initialization</b>				
Design and implementation of the database schema to manage the data in a MongoDB database.	10			
Populate the database with the data from the json files.	5			
Repository Implementation to read/write data from/to MongoDB	12			
<b>Complete and correct implementation of the requirements using Angular and Web API:</b>				
 Login and Navigation Bar	5			
 Get Tasks	8			
 Add Task	10			
 Update Task	5			
 Delete Task	2			
 Tasks Calendar	8			
 Sync Tasks to Google Calendar	0			
 Get Comments	7			
 Add Comment	8			
 Workload Summary Report	5			

<b>Documentation</b> - Compare and reflect on differences between the MVC-based design and implementation submitted in phase 2 with SPA-based design and implementation submitted in this phase. - SPA architecture diagram - Database schema diagram	10			
<b>Testing documentation</b> with evidence of correct implementation using <u>Unit Testing of Repositories and</u> snapshots illustrating the results of testing (you must use the provided template).	5			
<b>Total</b>	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.