HalaqaMetrash Web App

CMPS 356 Project Phase 2 – Web UI Implementation using client-side JavaScript and Data Management using MongoDB (15% of the course grade).

The project phase 2 submission and demo are due by 12pm Sunday 10/05/2020.

1. Deliverables

You are to complete the implementation of the HalaqaMetrash to deliver the use cases documented in the project phase 1 requirements. Your project phase 2 deliverables include:

Part 1 - Database Design and Implementation

1. Design and implementation of the database schema to manage HalaqaMetrash data in a MongoDB database. The implementation should use mongoose library.
2. Document your database schema design in a schema diagram.
3. Populate the database with the data from the json files.

Part 2 - Web Application Implementation

Implement client-side UI to deliver HalaqaMetrash use cases based on the previously developed and validated design. The UI design is provided in the html pages included in the base solution you will be using to implement the design.

The expected deliverables are summarized below:

1. Implement the *repositories* to read/write entities using 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., using aggregate query to get the data for the Halaqa summary report).
   - Implement the Web UI using Client-side JavaScript.
     - The same form should be used for add/update entities such as Task.
     - All dropdowns should be dynamically filled with reference data returned by Web API.
     - Once the form is submitted successfully, the app should redirect the user to the main page.

Part 3 – Design and Testing Documentation

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 learnt when you compare your submission with the model solution.
2. Database schema diagram
3. Testing document including screenshots of conducted tests illustrating a working implementation of both the Web API and the Web UI.

4. Every team member should submit a description of their project contribution. Every team member should participate in the solution demo and answer questions during the demo.

Important notes:

- Continue posting your questions to https://piazza.com/qu.edu.qa/spring2020/cmps356/

- Do not forget to submit your design and testing document (in Word format) and fill the *Functionality* column of the grading sheet provided in phase 2 Word template.

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

- You need to test as you go!

- Seek further clarification about the requirements/deliverables during the progress meeting with the instructor.

2. Grading rubric

Criteria	%		Quality of the implementation	Score
1) Database Design and Implementation	**10%**			
Design and implementation of the database schema to manage HalaqaMetrash data in a MongoDB database.	7			
Populate the database with the data from the json files.	3			
2) [45%] Implement the Web UI using Client-side JavaScript. Plus customize the application UI and behavior based on the user's role.	**45%**			
• Add parent	9			
• Add student	10			
• Get tasks (parent view)	10			
• Get tasks (teacher view)	15			
• Add Task	10			
• Update task	12			
• Complete task	8			
• Delete task	6			
• Get messages	8			
• Halaqa Summary Report	12			
3) [30%] Repository implementation to read/write data from/to MongoDB	**30%**			
• Login	10			
• Add parent	10			

• Add student	10		
• Get tasks	12		
• Add Task	10		
• Update task	8		
• Complete task	8		
• Delete task	5		
• Add message	5		
• Get messages	8		
• Halaqa Summary Report	14		
4) [15%] Design and Testing Documentation. *** [5 %] Design documentation:** - 5 lessons learned from Phase 1 - Database schema diagram *** [10 %] Testing documentation:** using screen shots illustrating the testing of Web UI and Web API (you must use the provided template).	**15%**		
Total	100%		
- Discussion of the project contribution of each team member [-10pts if not done]			
- Demo and discussion [-20pts if not done]			
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 Single Page Architecture (SPA)**, 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.