



# HalaqaMetrash Web App

## CMPS356 Project Phase 1 – Web UI Design and Web API Implementation

This is a group project worth 15%. The project submission is due by midnight Sunday 12<sup>th</sup> April 2020.

### 1. Requirements

Qatar is blessed by many Quran classes for kids. You are requested to design and implement a HalaqaMetrash Web App to allow parents to follow-up the progress of their kids and help teachers engage parents and easily communicate with them. The key use cases to deliver are shown in Figure 1.

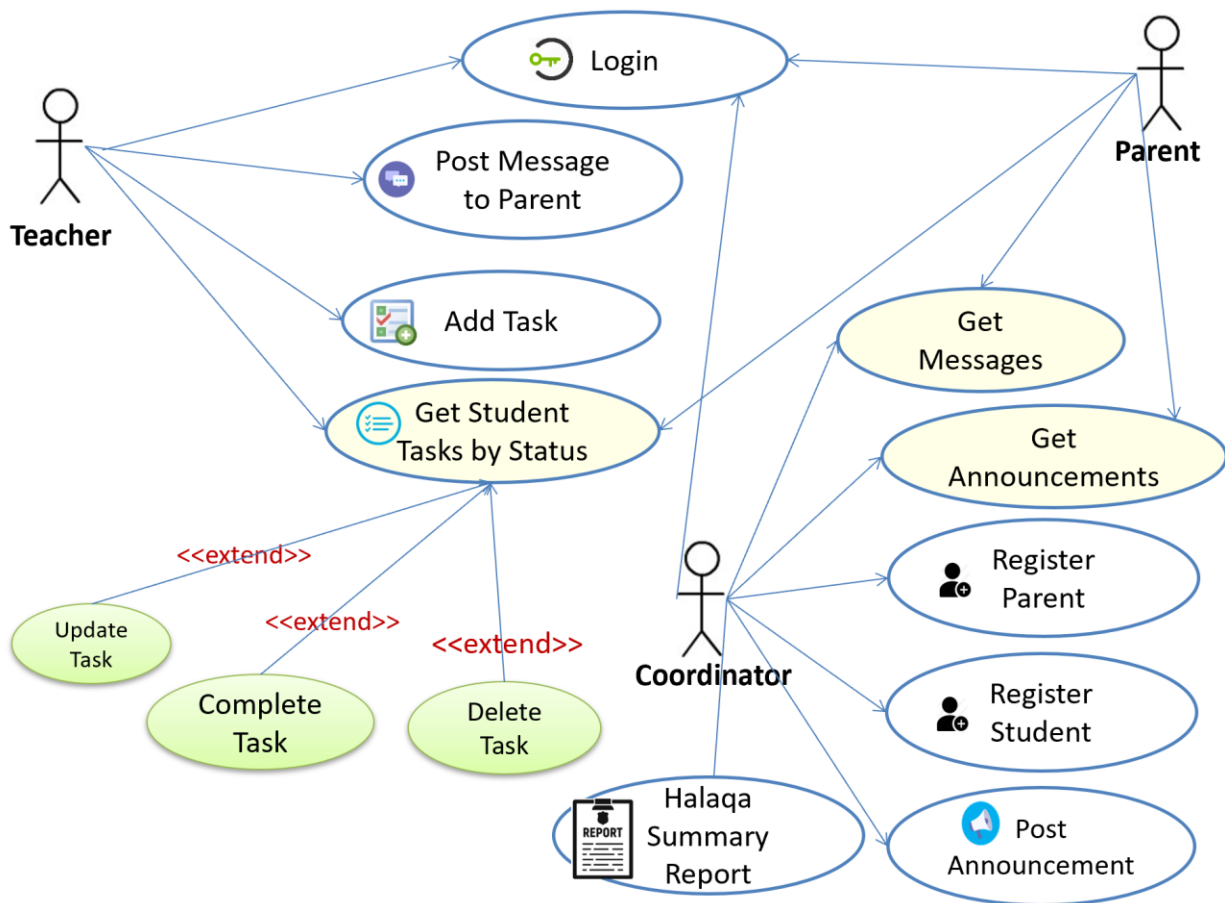



Figure 1. HalaqaMetrash Use cases

 **Login:** Allows the user (i.e., Coordinator, Teacher and Parent) to login to use the application using their email and a password.



**Register Parent:** The *Coordinator*, responsible of managing all Halaqat, can register parents. The registration should include the following parent details: first name and last name, Qatari Id, Mobile, Email and Password.



**Register Student:** The *Coordinator* can register 1 or many children for a parent. A child details include: First name, Last name, Date of Birth, Gender, and School Grade. Upon registration, the child is assigned to a Halaqa. Each Halaqa has a name and a teacher. The list of teachers and their Halaqa and the coordinator details are provided to you. But you still need to include these in your design.



**Manage Memorization and Revision Tasks:** The *Teacher* can assign memorization and revision tasks to each child to keep track of their progress of memorizing new Ayats or revising previously learned ones. It includes the following use cases:

- **Get Tasks** returns completed, pending or all tasks for a student. The parent can only access the tasks for their children. The teacher can access the tasks of students in their halaqa.  
For Pending Tasks, the teacher can either Complete, Update or Delete a task.
- **Add Task** allow the teacher to assign a task to a student. First the teacher selects the Sura, the Aya range, the due date (by default this should be set to Today's date + 1) and the type of task (Memorization or Revision). The Sura list and possible Aya range should be provided to the user to avoid typing them. The list of Quran Surahs is provided to you.
- **Update task** to update the task details (Sura, the Aya range, the due date and the type of task (Memorization or Revision)). This use case is similar to add.
- **Complete task**, the teacher can mark the task as completed then specify the Completed date (by default it should be set to Today's date), the Hifz level (Excellent, Ok, Poor) and optional comment.
- **Delete task** to delete a task.

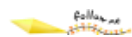


**Messaging:** The *Teacher* can post messages to parents informing them about the Halaqa learning achievements or child positive/negative behavioural or simply share that unforgettable Halaqa happy moments! The teacher can attach an image to their message such as photos from the Halaqa.

The parent can get messages for a child or all children.



**Announcements:** The *Coordinator* can post announcements informing them of important events such as Competitions and Holidays. They can allow attach an image to their announcement.



**Student Follow-up:** *Parents* can login and get the progress of the tasks assigned to their children. Also, they can get the messages sent by the Teacher as well as the announcements made by the Coordinator. Parents can only see the progress and messages for their registered children.

The Coordinator can also follow-up the progress and access the messages for any student.



**Halaqa Summary Report** returns for a date range (by default from 1<sup>st</sup> of the month to current date) the list of halaqa, the name of responsible teacher, students count, average count of revised ayat, and average count of memorized ayat.

## Deliverables:

- 1) Classes Design and the UI design to deliver HalaqaMetrash use cases.

The design documentation should include at least the following:

- Class Diagram showing Entities, Repositories and Services.
- 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) Design and implement the Web UI using HTML 5 and CSS. The pages should comply with Web user interface design best practices. Also remember that 'there is elegance in simplicity'.
- 3) Implement the repositories and the services (i.e., Web API) using JavaScript and Node.js to deliver HalaqaMetrash use cases based on your previously developed and validated design documented in deliverable 1. The application data should be managed using *json* files.

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

## 2. Grading rubric

Criteria	%	Functionality*	Quality of the implementation	Score
<b>1) [20%] Application Design</b> The design documentation should include at least the following: <ul style="list-style-type: none"><li>• Class Diagram showing Entities, Repositories and Services.</li><li>• Discussion of design rationale of 5 key design decisions.</li></ul>	20%			
<b>2) [30%] Design and implement the Web UI using HTML and CSS for:</b>	30%			
• Main page design (including menu)	5			
• Login	5			
• Register parent	8			
• Register student	8			
• Get tasks	10			
• Complete task	6			
• Delete task	5			
• Add Task	10			

• Update task	5			
• Post message	6			
• Get messages	8			
• Post announcement	6			
• Get announcements	8			
• Halaqa Summary Report	10			
<b>3) [40%] Implement the repositories and the services (i.e., Web API) using JavaScript for:</b>	40%			
• Login	6			
• Register parent	8			
• Register student	8			
• Get tasks	12			
• Complete task	6			
• Delete task	6			
• Add Task	10			
• Update task	6			
• Post message	6			
• Get messages	8			
• Post announcement	6			
• Get announcements	8			
• Halaqa Summary Report	10			
<b>4) [10%] Testing documentation</b> using screen shots illustrating the testing of Web UI and Web API.	10%			
<b>Total</b>	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**.

### 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

### Suggested Add/Update Task UI:

The image shows a UI design for an 'Add/Update Task' form. It includes a dropdown for 'Sura', two numeric input fields for 'From Aya' and 'To Aya', two date input fields for 'Due Date' and 'Completed date', and a 'Task Type' section with two radio buttons. Red callout boxes provide specific requirements for each field.

**Sura**  
2. Al-Baqara (286 Aya)

**From Aya:** 1

**To Aya:** 1

**Due Date**  
dd / mm / yyyy

**Completed date**  
dd / mm / yyyy

**Task Type:** ☐ Memorization ☐ Revision

**Annotations:**

- Whenever a Sura is changed the page should reset the *From Aya* and *To Aya* to 1 and it should auto set their max number.
- To Aya should be less than or equal From Aya.
- By default the Due Date should be set to Today's date + 1. It should be validate to ensure it is greater than or equal today.
- By default the Completed Date should be set to Today's date. It should be validate to ensure it is greater than or equal the Due Date.

Figure 2. Add/Update Task UI Design

### Resources:

- Quran Surah list <http://erradi.github.io/json/surah.json>
- Student list <http://erradi.github.io/json/student.json>
- Teacher list <http://erradi.github.io/json/teacher.json>
- Sample Task list <http://erradi.github.io/json/task.json>