HalaqaMetrash Web App

CMPS 356 Project Phase 1 – WebApp Design and Implementation

This is a group project worth 15%. The project submission is due by midnight Sunday 13th November 20202.

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.

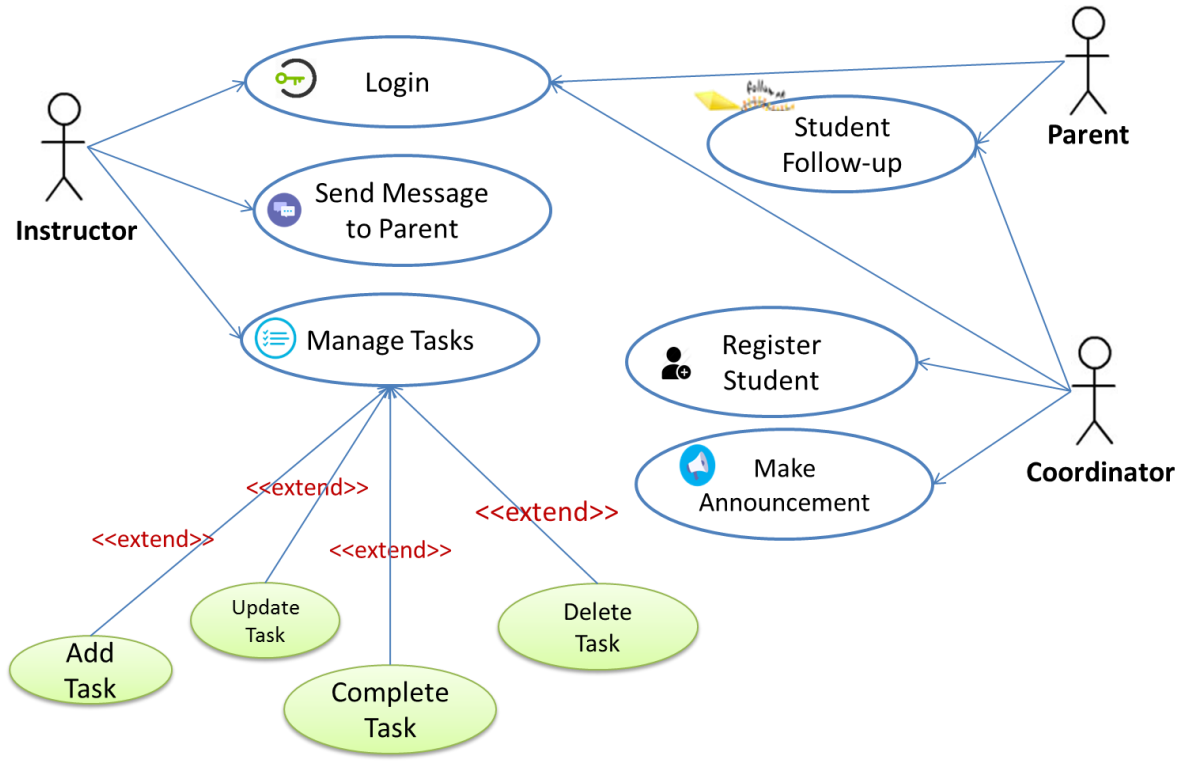

Figure 1. HalaqaMetrash Use cases

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

Student Registration: The *Coordinator*, responsible of managing all Halaqat, can register students. The registration should include the following minimum details: Parent first name and last name, Qatari Id, Mobile, Email, Username, and Password. A parent can register 1 or many children. 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. This user case includes the following 3 sub use cases:

- **Add Task** to 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.
- **List Tasks** to display either the completed or pending tasks for a particular student. For Pending Tasks, the teacher can either Complete, Update or Delete a task.
- **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.
- **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.
- **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.

Announcements: The *Coordinator* can broadcast messages to all parents at once 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 see the progress of their children. Also they can see 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.

Deliverables:

1) Architecture Design, Classes Design and the UI design to deliver HalaqaMetrash 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.

   During the weekly project meetings with the instructor, you are required to present and discuss your design with the instructor and get feedback.
2) Implement the client-side and the server-side Web components to deliver HalaqaMetrash use cases based on your previously developed and validated design.

The HalaqaMetrash should be fully implemented using React.js and Next.js. The application data can be managed either using *json* files or a database such as MongoDB. HalaqaMetrash web pages should use

React Components, HTML 5 and CCS. 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 HalaqaMetrash 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.	20		
Complete and correct implementation of the requirements:			
• Login	4		
• Student Registration	10		
• List Tasks	8		
• Complete task	8		
• Delete task	5		
• Add Task	9		
• Update task	5		
• Messaging	6		
• Announcements	5		
• Student Follow-up (Parent/Coordinator)	14		
Testing documentation with evidence of correct execution 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**: ***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 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.
- 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:

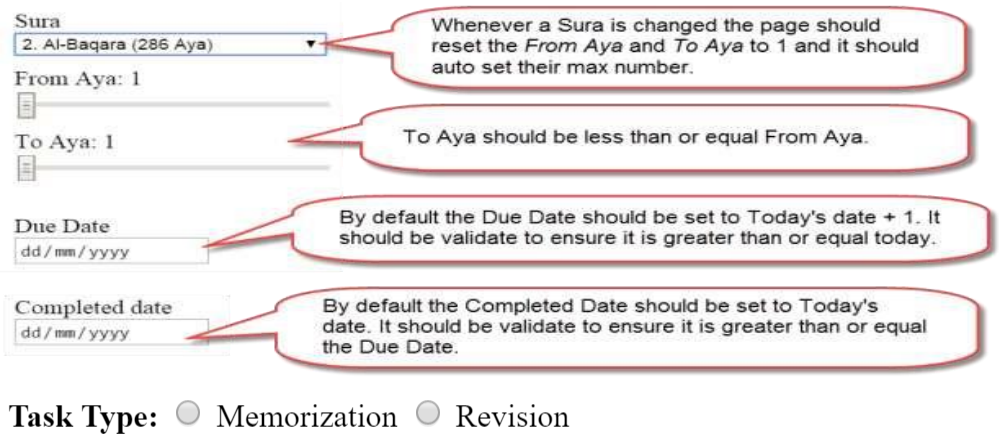

Figure 2. Add/Update Task UI Design

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