



School Management System (SMS) App

CMPT 661 Project Phase 1 – WebApp Design and Implementation



This is a very early draft to give you an idea about the project will look like. A complete version will be posted before Thursday's class.

The project submission is due by midnight TBD.

1. Requirements

You are requested to design and implement the School Management System (SMS) Web App to allow the school staff and parents to follow-up the progress of their kids and help teachers engage parents and easily communicate with them.

The main SMS modules to be designed and developed:

-  **Login:** Allows the user (i.e., *Principal, Teacher and Parent*) to login to use the application.
-  **Admissions Management:** Provide online forms for applications, then track each applicant step by step with through the entire admissions process. The system assigns a unique Student Identifier. 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. School staff can record entry test results.
Allow parents to query the status of their enrolment application. Change the application status to accepted, rejected or waiting list. Generate notifications to keep parents, applicants and staff informed though each step of the admissions process.
- **Registration Management:** Manage registrations, send registration reminders and generate various management reports. Financial records to keep track of received and pending payments.
- **Attendance Tracking:** Record student and teacher absences (including leaves approved by the school). Generate reports such as average monthly absences.
- **Behavior Management:** Teachers/Principal document and track student disciplinary incidents, maintain related records to help improve discipline by ensuring that students are held accountable for their actions. Record information such as:
 - Type of incident
 - Location and time
 - Student and staff involved
 - Grade level
 - Teacher remarks
 - Interventions, follow-up actions / penalties
 - Notifications to parents

Notify parents of infractions. Make information available online for students, parents and staff. Instantly access any student's complete disciplinary history when speaking with parents.

Produce statistical reports including statistical analysis of discipline data to track trends.

- **Grade Center:** to manage and communicate grades to students and parents. Help in measuring and communicating achievements of individual student, whole class or all classes through visual graphs to the class level, student level, or assessment level.

- **School-to-home communication:**



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



Announcements: The *Principal* 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.

- **Parent Portal:** access to various reports about the above. Channel of communication with the school. This allows engaging parents with enhanced communication tools, making grades, attendance and more visible for all stakeholders.



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 Principal. The Principal can also follow-up access the messages for any student.

- **Integration with Google Classroom:**

- Create classes
- Enroll students
- Synch classes and students
- Import/Export grades to Google Classroom (if possible)

- **SMS API:** provide Web API to allow other applications to get the student details, absences and student grades.

The key use cases to deliver are shown in Figure 1.

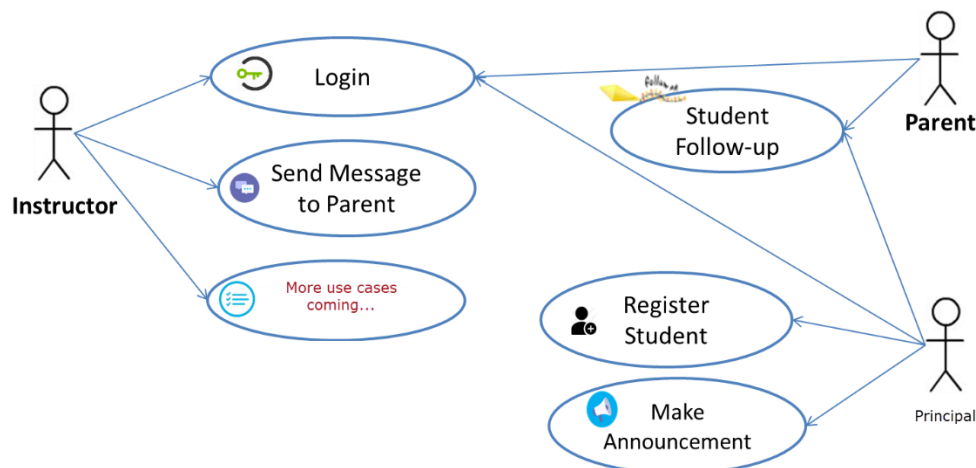


Figure 1. SMS Use cases (incomplete more coming soon)

Deliverables:

- 1) Architecture Design, Classes Design and the UI design to deliver SMS 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.
- 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) Implement the client-side and the server-side Web components to deliver SMS use cases based on your previously developed and validated design.

SMS should be fully implemented using Node.js. The application data can be managed either using *json* files. SMS web pages should use 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 SMS use cases. The design documentation should include at least the following: <ul style="list-style-type: none">• Application Architecture Diagram• Class Diagram showing Entities, Repositories and Services and Controllers.• UI Design and navigation.• Discussion of design rationale of key design decisions. | 20 | | |
| Complete and correct implementation of the requirements: | 0 | | |
| <ul style="list-style-type: none">• Login | 4 | | |
| <ul style="list-style-type: none">• Student Registration | 10 | | |
| <ul style="list-style-type: none">• ?? | 8 | | |

| | | | |
|----------------------------------------------------------------------------------------------------------------------|-----------|--|--|
| • ?? | 8 | | |
| • ?? | 5 | | |
| • ?? | 9 | | |
| • ?? | 5 | | |
| • Messaging | 6 | | |
| • Announcements | 5 | | |
| • Student Follow-up (Parent/Principal) | 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.
- 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.

4. Project Phase 2 and Phase 3

- Phase 2: enhance the app you built in phase 1 to a Single Page Application (SPA) using Angular. Data management using MongoDB instead of files.
- Phase 3: enhance the app you built in phase 3 to a Hybrid Mobile-Web App using Ionic.

More details and guidance will be provided.