School Management System (SMS) App

CMPS 350 Project Phase 1 – WebApp UI Design and Implementation (15% of the course grade).

The project phase 1 submission is due by midnight Sunday 21th March 2021. Demos during the same week.

1. Requirements

You are requested to design a School Management System (SMS) Web App for managing various school processes including admission, registration, grading and absence. The application will allow the school staff and parents to follow-up the students' progress and help teachers engage parents and easily communicate with them.

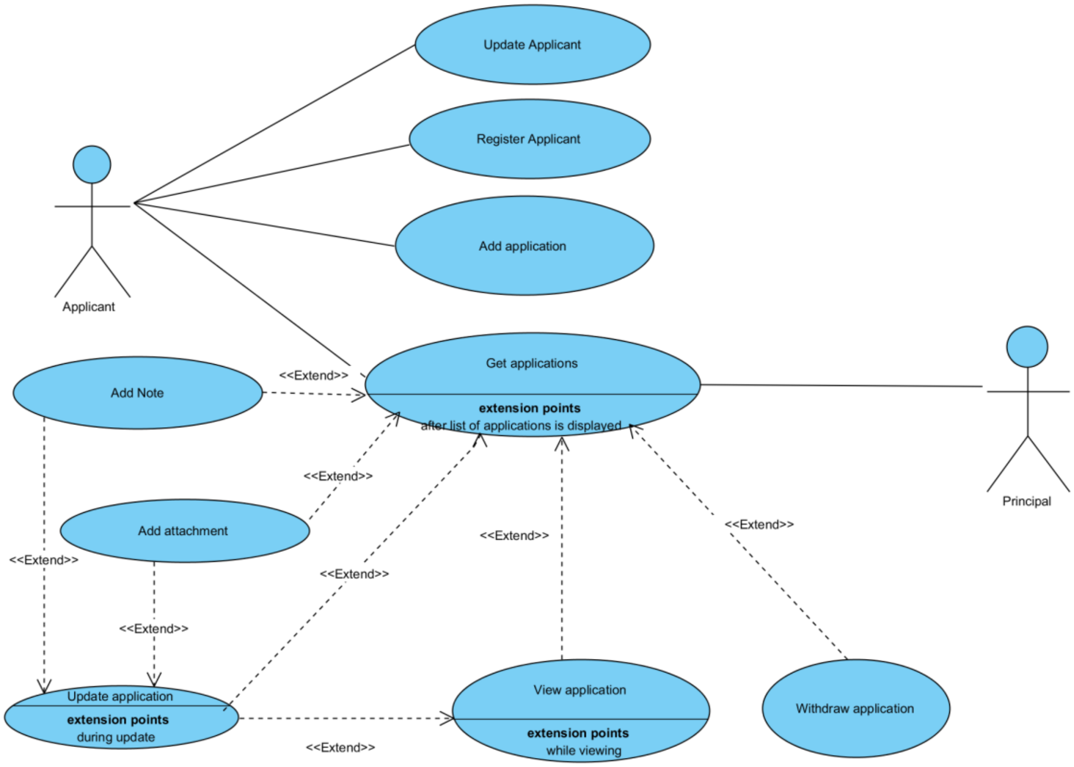
The main SMS modules to be designed and developed are described below.

All modules include 🔑 **Login** to allow the user (i.e., *Principal*, *Teacher* and *Parent*) to login to use the application.

1.1. Module 1 - Admission Management

This module should support the following use cases:

Use Case	Description
Register applicant	Allows the applicant to register an account
Update applicant	Allows the applicant to update his/her details
Add application	Allows the applicant to submit an application
Get applications	Returns a list of applications submitted by the applicant
View application	Allows the user to view a non-editable view of the application
Withdraw application	Allows the applicant to withdraw an application
Update application	Allows the user to update an application
Add attachment	Allows the user to add an attachment to an application
Add notes	Allow the user to add a note to the application
Applications summary report	Generate applications summary report showing the number of applications per grade and by application status (e.g., for a particular academic year, list for each grade list the number of received applications, the accepted, rejected, pending, waiting list, withdrawn).



- **Applicant Registration:** Allows applicants (typically parents) to register to be able to submit an Admission Application for one or many students to join the school. The registration should include the following minimum details: Father first name and last name, National Id, Home Phone, Mobile, Email, Occupation, Name of Employer (same details should be entered for the mother). The applicant can enter a password to be able to login later using their email and their selected password.
- **Admission Application**: Once registered, the applicant (i.e., parent) can submit an Admission Application for one or many students. For each application the student details must be entered including: First name, Last name, Date of Birth, Gender, Current School Grade, Grade applying for, Name of Current School. The applicant can also enter comments such as the fact that they are applying from overseas. Note that each student needs a separate application. The Admission Application should allow attaching documents such as previous school reports. Once submitted, the initial application status should be 'Submitted' and the system assigns it a unique Student Identifier. The application should be associated with the academic year flagged as open for admission (typically the next Academic Year).
- Update **Admission Application**: Upon login the system should show the list of available applications so that the applicant can either view (e.g., check Admission Application status), update or withdraw an application. If an application is withdrawn, then its status should be changed to 'Withdrawn' and the Principal and the Applicant should be notified.

  The applicant can also update and resubmit a withdrawn or rejected application. In such case, upon submission it should be auto-assigned to the academic year flagged as open for admission (not the initial academic year at the time it was first submitted).

The Applicant can view an application details including attachment and notes that flagged as visible to applicants.

Note that applications with status Accepted, Rejected or Waiting list can longer be updated.

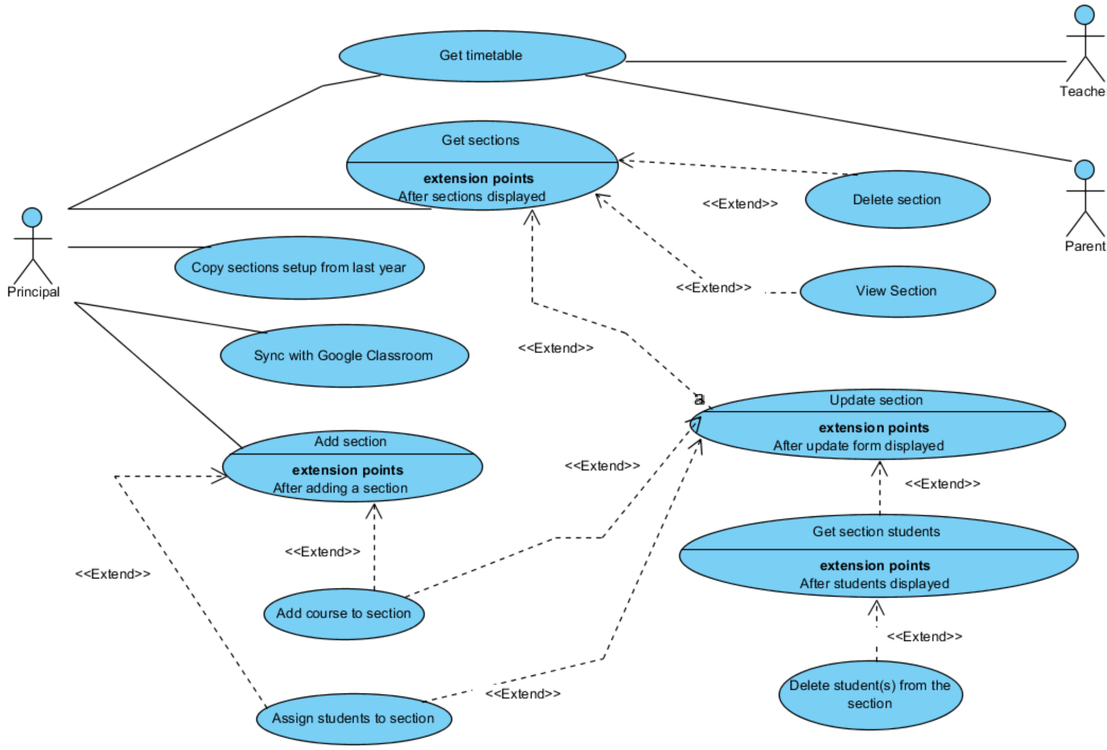
- The Principal should have the ability to perform the following:
  - Get the list of applications with the ability to filter by status (including get all) and by academic year (by default it should be set to the academic year configured with 'open for admission').
  - View an application details including attachment and notes ○ Update an application details including:
    - Change the application status to Pending, Accepted, Rejected, Withdrawn, or Waiting list.
    - Attach/Delete an attachment
    - Add/Update an Admission Test and its date. When an Admission Test is entered or updated then the applicant should be notified.
    - Add/Update Admission a Test Result and Comments (there could be many test results per application). When an Admission Test results are entered/updated then the applicant should be notified.
    - The applicant & coordinator should be able to view the test results associate with an application.
    - Add/Update a note such as a summary of conversation with parents, summary of discussion of the admission committee. The coordinator can decide whether the added/updated note can be visible to the applicants. By default, notes are visible only to the Principal.
    - View previous notes attached to an application.
    - Generate applications summary report showing the number of applications per grade and by application status (e.g., for a particular academic year, list for each grade list the number of received applications, the accepted, rejected, pending, waiting list, withdrawn). ~~Comparative reports of admissions for the last 3 academic years with graphs.~~

Note that an applicant can have one or many students. Each student can have one or many admission applications. A student can have the following admission status: pending, accepted, rejected, withdrawn, or waiting list. Student has first admission academic year property.

1.2. Module 2 – Sections and Schedule Management

This module provides use cases to maintain the following structure:

- Each Grade Level has one or many sections. The principal can add, update and remove sections for a particular grade.
- Each section has a set of courses scheduled at particular days and times and at a particular classroom. A section-course has an assigned teacher. A teacher can teach one or many courses.
- The principal should be able to assign teachers and classrooms to a section.
- The principal should be able to assign students to a section.



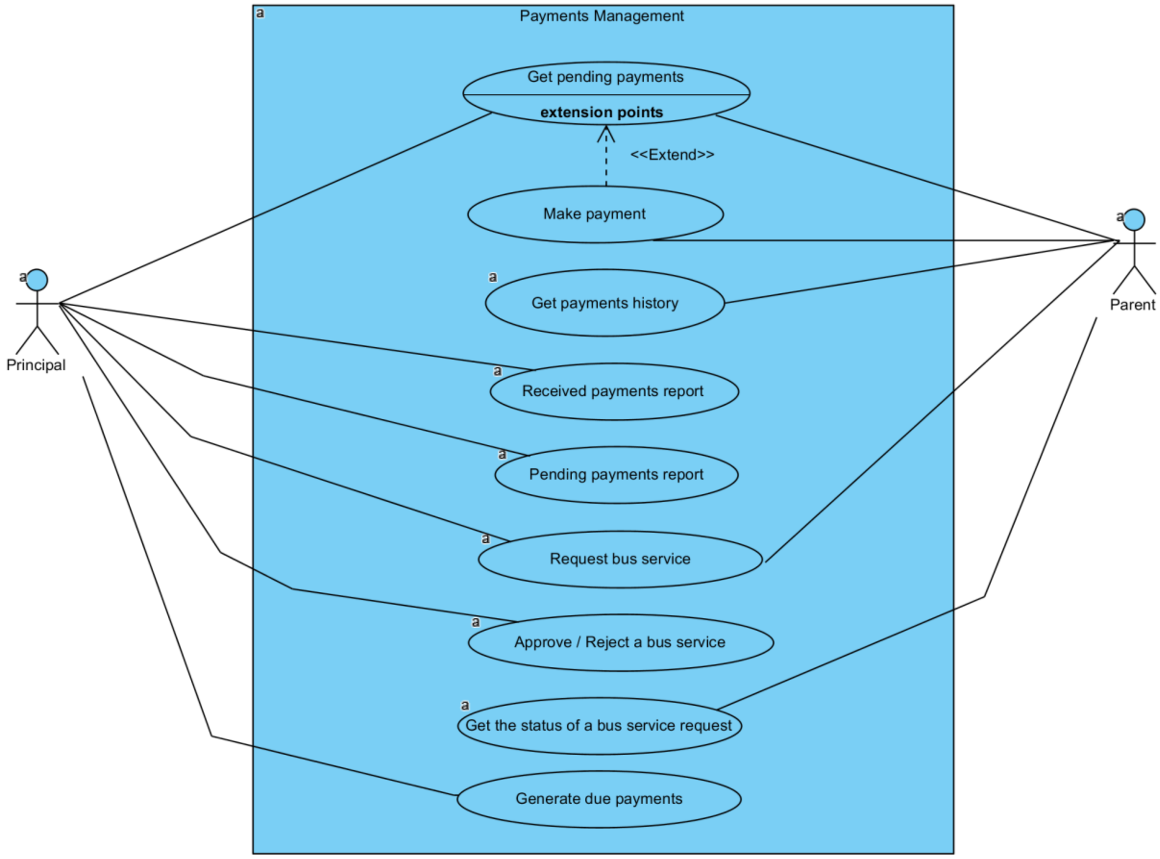
Use Case	Description
Get sections	Principal can get sections filtered by Academic year and Grade Level.
View section	View a section details including scheduled courses and enrolled students.
Add section	Principal is able to add a section. First select a grade level to add a section for it. It is not mandatory to add courses and students at the same time of adding a section.
Update section	Principal is able to update a section and associated courses list and students list.
Delete section	Principal is able to delete a section and associated courses list and students list.
Assign students to section	Principal is able to select one or more available students having the same grade level as the section and assign them to the selected section.
Get section students	Get the students enrolled in a section.
Delete student(s) from the section	After getting the list of section students then the principal can delete one more students from the section.
Add course to section	Add, update or delete a course to a section. The principal can specify the course schedule (days and times), the assigned classroom and the assigned teacher.
Copy sections setup from last year	Instead of starting from scratch the principal should be able to copy the sections setup from previous year then make the required modifications.
Get timetable	• Principal can view the timetable of the teachers and the timetable of any section. • Teachers can view their own timetable. • Parents can view their kids timetable. First the parent can select a student to get their timetable. If they have only one student then the timetable should be

	displayed directly.
Sync with Google Classroom **(to be done in phase 2)**	Create classes, enroll students, and sync classes and students with Google Classroom.

1.3. Module 3 - Payment Management

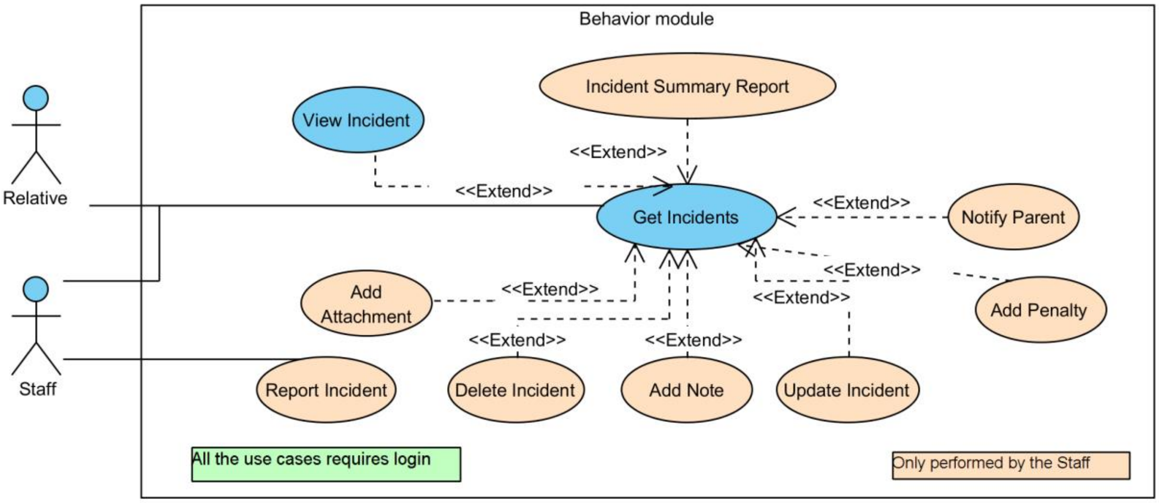
This module delivers the following use cases:

Use case	**Description**
Setup and maintain the free structure	Setup and maintain the free structure per semester and per level (example 1 and example 2) including Tuition Fee, Registration Fee, Transport Fee, etc. The principal should be able to add/update different types of fees per level.
Get pending payments	Get pending Registration, Tuition of Bus fees of a student.
Make payment	Allow paying Registration, Tuition of Bus fees of a student. The system should record the payment date, the amount paid and payment mode (cash, bank card, credit card, cheque, direct bank deposit).
Get payments history	Get past payments for a student within a date range (by default the current academic year)
Pending payments report	Report of pending Registration, Tuition and Bus fees of a student or all students. The report should allow filtering by Fee Type. Also, it should provide a summary of pending payments per type.
Received payments report	Report of received Registration, Tuition and Bus fees of a student or all students within a date range (by default the current month). The report should allow filtering by Fee Type. Also, it should provide a summary of received payments per type.
Request bus service	Request a bus service registration for a student.
Approve / reject bus service request	Get the list of pending bus service requests. Approve / reject a request with ability to add a comment.
Get status of bus service request	Get status of bus service request for a particular student.
Generate due payments	Generate due payments for all active students.



1.4. Module 4 - Incidents Management

Using this module, 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. This will allow instant access to any student's complete disciplinary history when speaking with parents. Also, statistical reports can be produced including statistical analysis of discipline data to track trends. The use cases of this module are:



Use Case	Description
Get Incidents	Parent can only get his kid’s incidents. Principal can get all the incidents using different filters: - Incidents for a particular student or all students. - Incidents for a particular grade level. - Incidents within a date range.
Report incident	Principal add an incident. Record information such as: - Type of incident, Location and time, Student and staff involved - Grade level, Teacher remarks
Update Incident	Principal can change the information submitted to the system.
View Incident	View incident details including notes, attachments, and penalty.
Add Penalty	Principal can add a penalty to an incident.
Add attachment	Principal can add attachments to an incident
Add Note	Principal can add notes to an incident.
Incidents Summary Report	Report incidents per academic year. The report should provide the summary of incidents by type and the summary of incidents by grade level. With the ability to drill into the details. For example, the report could state that there were 6 incidents of ‘Damaging the school property’. Then the user can click into this summary to get the actual incidents.

2. Deliverables

Seek further clarification about the requirements/deliverables during the initial progress meeting with the instructor. Note that further important clarifications maybe modified/added to your assigned module.

During the weekly project meetings with the instructor, you are required to present and discuss your design with the instructor and get feedback.

1) Design the App Web UI and navigation.
2) Implement the app Web UI and navigation using HTML, CSS and JavaScript. The pages should comply with Web user interface design best practices. Also remember that ‘there is elegance in simplicity’.
3) Implement the client-side data access repositories to read/write the app data from/to IndexedDB.
4) Application design documentation including the Entities Class Diagram and the Repositories Class diagram.
5) Create test data JSON files for the entities of your module.
6) Document the app testing using screen shots illustrating the results of testing.

- Every team member should submit a description of their project contribution. Every team member should demo their work and answer questions during the demo.
- Push your implementation and documentation to your group GitHub repository as you make progress.
- Note that this phase will be focused only **a fully working client-side implementation** using data stored in json files and local browser databases. In phase 2 you will implement the server-side to move some of the computation and data management to the server-side.

3. Grading rubric

Criteria	%	Functionality*	Quality of the implementation
1) Design the App Web UI and navigation.	10		
2) Implement the app Web UI and navigation using HTML, CSS and JavaScript.	30		
3) Implement the client-side data access repositories to read/write the app data from/to IndexedDB.	40		
4) Application Design: Entities Class Diagram and Repositories Class diagram.	8		
5) Create test data JSON files for your module entities	6		
6) Testing documentation using screen shots illustrating the testing results. - Discussion of the project contribution of each team member.	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 60% 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. Solution quality also includes 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, unclean/untidy submission and unnecessary complex/poor user interface design.