School Management And Reporting Tool

Due Tuesday, April 13 at 8 a.m.

CSE 1325 - Spring 2021 - Homework #8 / Sprint 4 - 1 - Rev 1

Assignment Background

The School Management And Reporting Tool (SMART) will assist administrators of elementary and secondary schools with keeping track of students and their parents, teachers, course subjects, sections, and grades, and the many relationships between them.

Your goal is to prove that you can implement the SMART (and possibly smart) specification and thus win the contract to build the full production version along with the associated fame and cash and possible spacecraft tickets for Mars on-site support.

This is sprint 4 of a 5-sprint, 5-week project that you can add to your growing resume, with emphasis on expanding the class diagram.

Your implementation is permitted to vary from any class diagram provided with the final project as needed without penalty, as long as you correctly implement both the letter of and the intent of the feature list. If you prefer to build a tool that addresses a different cultural approach to educating children, that would simply be delightful. **If you have questions about the acceptability of changes that you are considering, contact the professor first!**

Sprint 4 - More Classes and a Template

The primary goal for sprint 4 is add additional supporting classes in preparation for tying the system together in the final sprint.

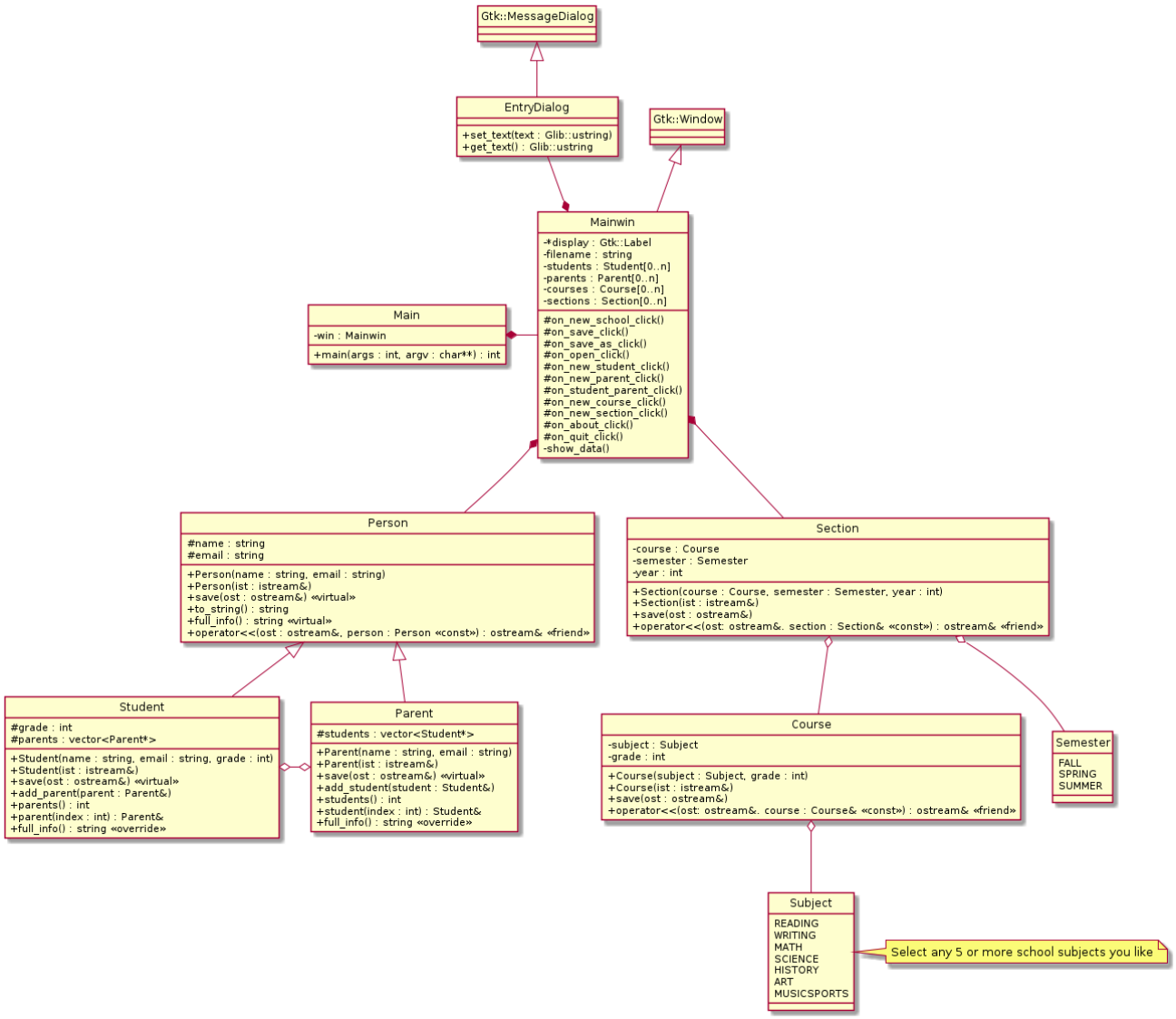
A suggested class diagram for the fourth sprint is on page 2. This diagram is split into multiple diagrams for easier reading, a very common approach as projects grow in size.

For this sprint, you **MAY** deviate from the class diagram as long as you fulfill the intent of the requirements, including all features allocated to this sprint in Scrum's Product Backlog.

For this sprint, you **MAY** adopt code from the sprint 3 suggested solution with the following caveats:

- Code from any class may be adapted under the terms of the Gnu General Public License 3 (GPL3), which requires attribution among other restrictions. The About dialog added in Sprint 3 is a good way to fulfill these obligations. **Remember to observe these license restrictions if you publish your project code after May 30 as part of your resume.**

Class Overview



EntryDialog, Main, Person, Student, Parent

These classes should require no changes for this sprint.

Semester and Subject

These are enum classes, which should be stored in separate .h and .cpp files. In addition to the enum class declaration, include

- **operator<<** function - Stream out the lower-case word matching the enum element (e.g., for Subject::READING, output "reading"). This can be used both for human output and for File > Save.
- **load_subject(ist)** and **load_semester(ist) functions**, respectively - These read a single word (with getline) from the input stream, and return the corresponding Subject or Semester element (respectively). This is used for File > Open.

It is sometimes convenient to create a const std::vector of an enum containing exactly one of each enum member, in order, to simplify iterating over all of the enum members in order with a for-each (since C++ inexplicably doesn't provide this capability). I did this with Subject, but not the smaller Semester. It's your call.

Course

A course is similar to CSE1325, but in elementary and secondary would look like "English (grade 4)". The data is defined in the constructor, accessed for output via operator<<, and saved and restored with the usual Course(ist) and save(ost) members.

Section

This is similar to Course, but with 4 attributes rather than 2 attributes.

Mainwin

Mainwin adds courses and sections attributes, most likely vectors, and on_new_course_click() and on_new_section_click() methods in which the data is obtained from the user. Make creating these objects available from both the Insert menu and the toolbar (2 more icons!). If a data type (student, parent, course, or section) is successfully created, it would be convenient to switch the view to display all of that data type. It's fine to use EntryDialog for creating the new objects in the callbacks, but of course unified dialogs with drop-downs and spinners look nicer.

Provide a mechanism to display sections or courses rather than students and parents in the main window display area. The usual convention would be a new menu named View, with View > Courses, View > Sections, View Students, and View Parents. Other acceptable options include Toolbar buttons or tabs.

Also update Mainwin::on_save_click and Mainwin::on_open_click to handle the new vectors, of course. Since we're adding new data types to the file format, here's an (optional) opportunity to provide backward compatibility with the older format. Not as easy as it looks!

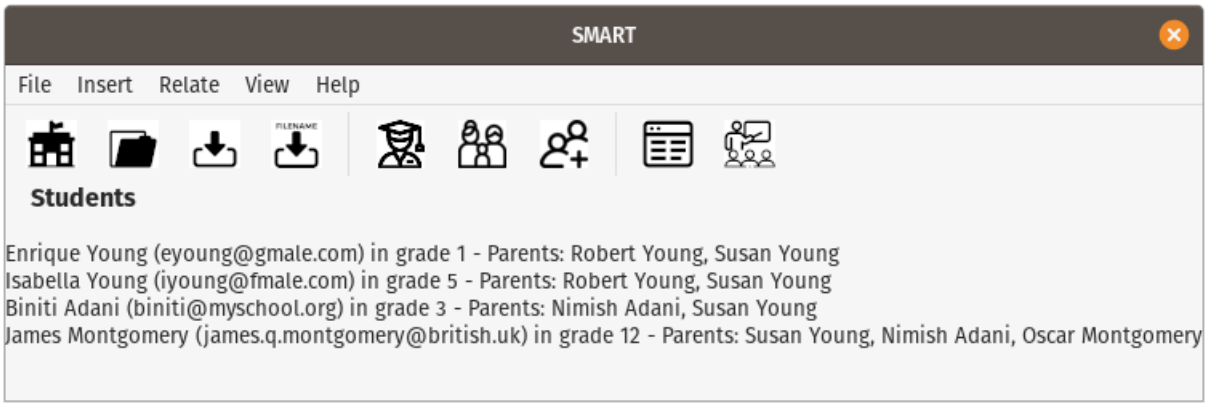
Makefile

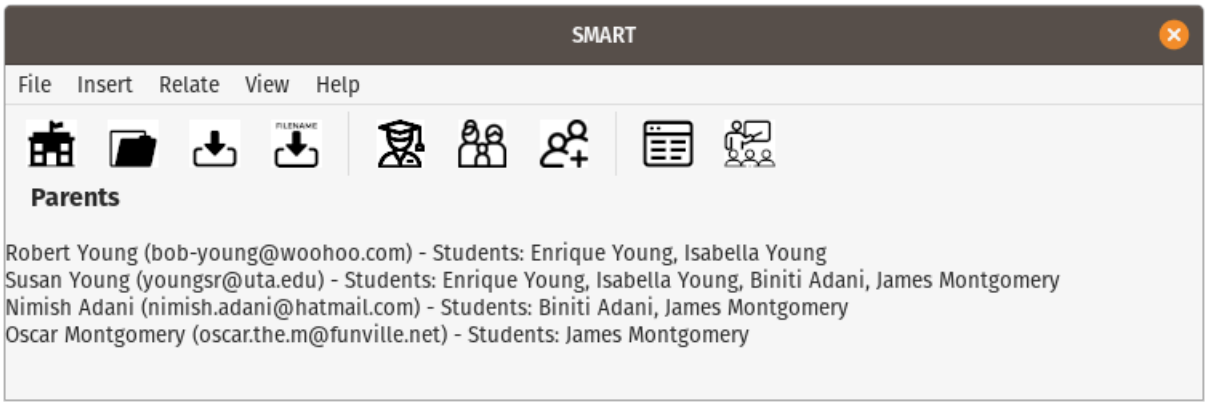
You will need to accommodate building the 2 new classes and 2 new enum classes this week. Since the template is only in a header file, it is not represented in the Makefile at all.

Screenshots

Below are some screenshots from the suggested solution. **Your application should look different than these,** since your logo and some icons should be custom. Your application may in fact look much *better* than these. Just ensure you meet all of the required features for this sprint!

Here are the four views, selected (in the suggested solution - *your implementation may vary*) from the View menu.





SMART

File Insert Relate View Help

Courses

reading (grade 1)
writing (grade 2)
math (grade 3)
science (grade 1)
history (grade 2)
art (grade 3)
music (grade 1)
sports (grade 2)
reading (grade 3)
writing (grade 1)
math (grade 2)
science (grade 3)
history (grade 1)
art (grade 2)
music (grade 3)
sports (grade 1)
reading (grade 2)
writing (grade 3)
math (grade 1)
science (grade 2)
history (grade 3)
art (grade 1)
music (grade 2)
sports (grade 3)
reading (grade 1)
writing (grade 2)
math (grade 3)
science (grade 1)
history (grade 2)
art (grade 3)
music (grade 1)
sports (grade 2)

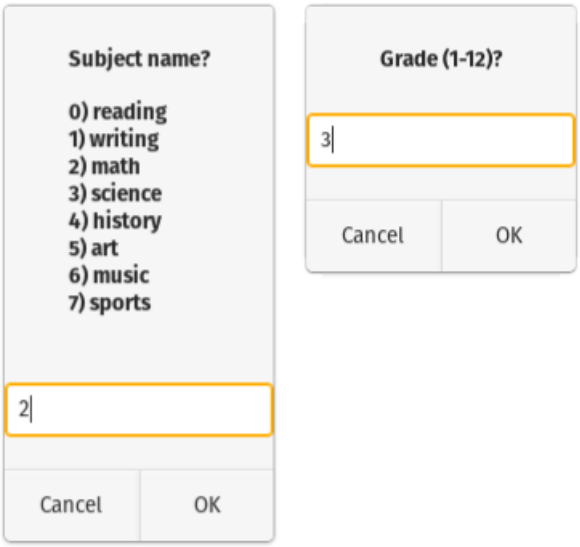
SMART

File Insert Relate View Help

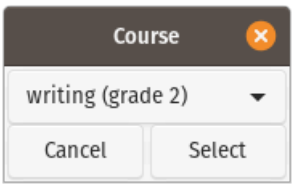
Sections

reading (grade 1) for fall 2021
math (grade 3) for fall 2021
music (grade 3) for fall 2021
history (grade 2) for fall 2021
art (grade 2) for fall 2021
reading (grade 2) for fall 2021
history (grade 1) for fall 2021
art (grade 1) for fall 2021
reading (grade 3) for fall 2021
writing (grade 3) for fall 2021
science (grade 1) for fall 2021
art (grade 2) for fall 2021
music (grade 1) for fall 2021
science (grade 1) for fall 2021
reading (grade 2) for fall 2021
reading (grade 2) for fall 2021
music (grade 1) for spring 2021
music (grade 3) for spring 2021
art (grade 3) for spring 2021
math (grade 3) for spring 2021
math (grade 3) for spring 2021
history (grade 2) for spring 2021
sports (grade 2) for spring 2021
sports (grade 3) for spring 2021
sports (grade 2) for spring 2021
history (grade 1) for spring 2021
science (grade 1) for spring 2021
math (grade 3) for spring 2021
music (grade 1) for spring 2021
writing (grade 2) for spring 2021
reading (grade 1) for spring 2021
music (grade 1) for spring 2021
science (grade 2) for summer 2021
music (grade 3) for summer 2021
math (grade 2) for summer 2021
reading (grade 1) for summer 2021
sports (grade 2) for summer 2021
music (grade 2) for summer 2021
music (grade 2) for summer 2021
sports (grade 2) for summer 2021
reading (grade 3) for summer 2021
writing (grade 3) for summer 2021
history (grade 3) for summer 2021
music (grade 3) for summer 2021
art (grade 1) for summer 2021
history (grade 2) for summer 2021
sports (grade 2) for summer 2021
science (grade 1) for summer 2021

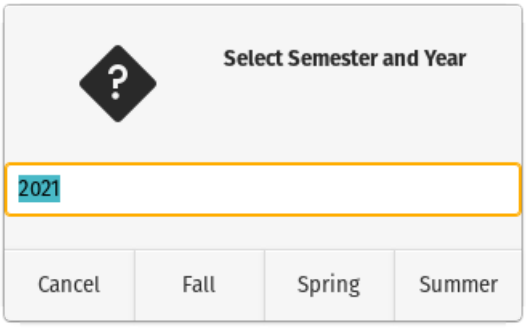
The two dialogs for creating a new Course. A ComboBoxText and an Entry would fit these into a single, more convenient dialog (for bonus points on Sprint 5).



The two dialogs for creating a new Section are below. The first is slightly more usable than an EntryDialog with a text menu, with a Gtk::ComboBoxText holding the courses. Create the EntryMessage version first, though, to have something working by the end of the sprint, before getting more sophisticated.



The second dialog is just an EntryDialog with add_button for Fall, Spring, and Summer. This already looks rather nice. A Gtk::MessageDialog with the above Gtk::ComboBoxText, then an Entry for the year, and finally buttons for the semester, would make a simple but nice custom dialog, no?



If you have questions about any part of this project, contact me via email or Teams. Ask questions. I **love** questions!