Justin Farnsworth

Assignment 3 – Open Source Software: Proposal and Specifications

VM: csc-415-server08.hpc.tcnj.edu

GitHub : https://github.com/farnswj1/ProjectSCHEMA

Option Choice

For this project, I will be choosing Option 3, which is to extend the functionality of the course enrollment application from Assignment 1. I am interested in adding more features to it and I also feel confident that I can create a perfectly functional application by the end of the semester.

Project Name

This project name will be called Student Course Help & Enrollment Management Application, or SCHEMA.

Problem

The enrollment application from Assignment 1 was a prototype and therefore not perfect. More work must be done to address any bugs or other unwanted results from the original submission. Also, the app is limited to only enrolling students. It lacks other essential features such as allowing a student to drop a course or allowing an administer to enroll a student into another course. There is also no waitlist for students who wish to enroll in a course that has no available seats.

Functionality

For each type of user, the application will include, but is not limited to:

- Students
    - Enroll - students can enroll in their desired courses as long as they satisfy the prerequisites for the courses.
    - Drop - students can drop courses if they no longer want to be enrolled in a course.
    - Add to waitlist - students can place themselves on a waitlist for a course.
    - Read class description - students can read information about the courses.

- Professors
    - Enroll students - a professor can enroll a student in a course.
    - Add/remove student from waitlist - a professor can remove a student from the waitlist.
    - Read class description - students can read information about the courses.
    - Write class description - students can write/modify information for the course.
- Administrators
    - Enroll students - an administrator can enroll a student in a course.
    - Add/remove student from waitlist - an administrator can remove a student from the waitlist.
    - Designate professor - an administrator can select a professor to teach a specific course/section. This function will also allow the administrator to change the professor teaching it.
    - Create course - an administrator can create a course and/or course sections for the students to enroll in.
    - Remove course - a professor can remove a course or course section if not enough students enroll in it.

The following may be tentative, however I may add a login feature that allows students, professors, and administrators to access SCHEMA.

Data Structures

The student, professor, and administrator data types will be held in their own hash tables. Their IDs will be their respective hash keys. This eliminates having to search through entire lists just to find the data type needed, sparing us time and energy. A database may be created too as a backup and to store all the information.

Algorithms

By using hashes, the time needed for searching is greatly reduced. The student classes will save the course IDs and course sections they are taking, allowing us to quickly find the course(s) data structures they are already in. Should a database be used, the primary keys will be the IDs of that respective class. For example, the primary key for a course will be the course ID and for the student a student ID. Since students can be enrolled in multiple courses, a link table will be needed as well.

To ensure that the course sections are balanced while enrolling students, modular arithmetic (long division) will be used. The total number of students across all sections for a particular course will be divided by the total number of sections. The remainder will be the section they will be enrolled in.

It is also important to allow students, professors, and administrators to access their respective profiles without worrying about others accessing it. More importantly, passwords must be kept secret and protected. To add some security, I tentatively plan on adding cryptographic hash functions, notably SHA-512, to encrypt the passwords before saving them. This way, anyone who tries to obtain the passwords will not be able to decipher it.

Implementation

This project will be a web-based application and written using Ruby and Ruby on Rails. Since Ruby on Rails also supports databases, I can use this to store information about the students, professors, courses, and administrators. I may also need to use SQL to create the databases as well.

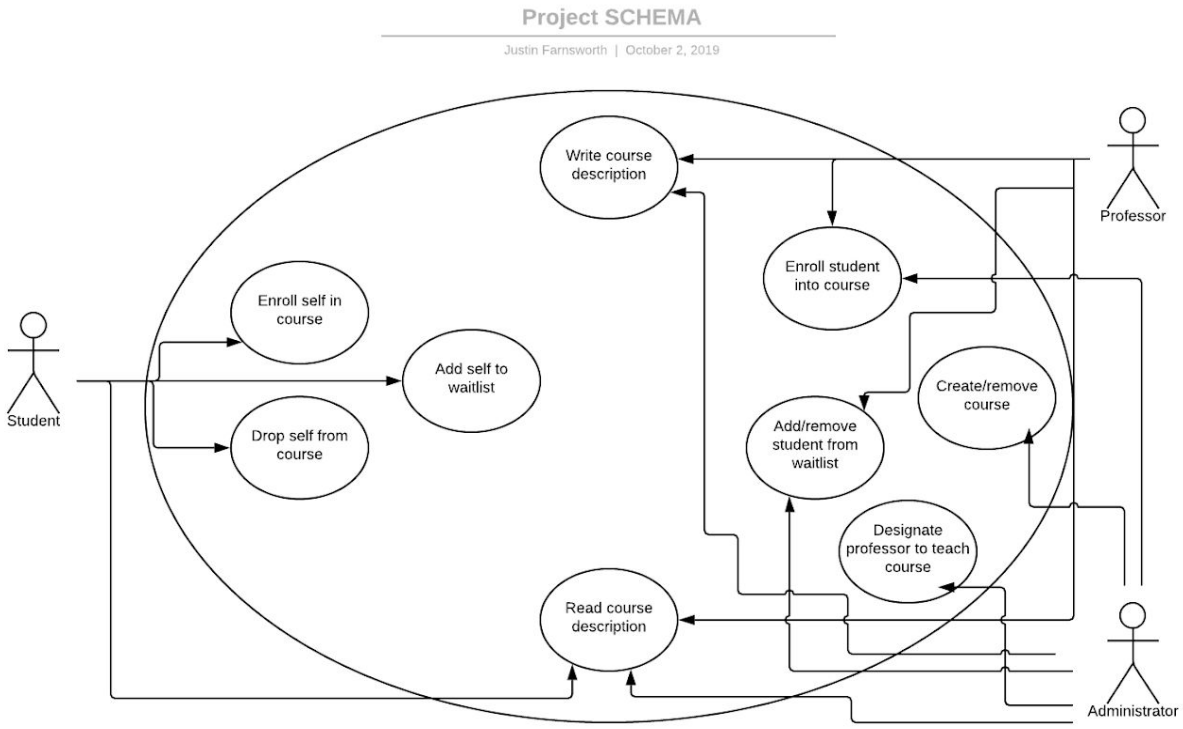
Challenges

While I have a fair understanding of Ruby, it will simply not be enough. I will need to learn Ruby on Rails and possibly review SQL. I will also need to refresh my understanding of databases.

Concepts Learned/Reinforced

I want to approach this problem using the evolutionary model as I want to test this application frequently. I also want to ensure that if any changes have to be made, it will be much more manageable. I also want to utilize sequence diagrams and state diagrams as they can greatly help with planning.

Use Case Diagram



Projected Timeline

October 3 - Outline drawn and abstraction takes place.

October 11 - Expand list of functionalities and include non-functional requirements.

October 18 - Course and student classes are created and tested.

October 25 - Professor class and administrator classes are created and tested.

November 1 - Database is created.

November 8 - Interface class is created and tested.

November 15 - Develop website.

November 22 - Testing and debugging.