GROUP 10: MILESTONE 4

Design Report

CampusCentral

Group Members:

Name: Karanvir Basson
Student ID: 1129746
Email: kbasson@uoguelph.ca

Name: Andrew Chow
Student ID: 1088114
Email: achow04@uoguelph.ca

Name: Fee Kim Ah-Poa
Student ID: 1159794
Email: fahpoa@uoguelph.ca

Name: Mazen Bahgat
Student ID: 1157821
Email: mbahgat@uoguelph.ca

Name: Muhammad Talha Sadaqat
Student ID: 1145327
Email: msadaqat@uoguelph.ca

Name: Hadi Rana
Student ID: 1107555
Email: hrana04@uoguelph.ca

Name: Joshua Jones
Student ID: 1125364
Email: jjones21@uoguelph.ca

Name: Cavaari Taylor
Student ID: 1159034
Email: cavaari@uoguelph.ca

Project Redesigns:

Removed Course Waiver Functionality:

Originally, we decided to implement waiver features in the course schedule such that a user can request a waiver for a course that they are not originally qualified to take. We decided to remove this feature for the sake of simplicity, and instead have all waiver request communications take place externally (through Gryphmail or in-person communication for example).

Renamed “Course” Class to “CourseOffering”:

Originally, the “Course” class represents an offering of a course in a specific semester, and thus contains all of the relevant information that applies to the specific offering of a course in its instance variables. We decided to change the name of this class from “Course” to “CourseOffering” in order to further specify that this object takes the general course information and applies it to the specific offering for a given semester.

Renamed “CourseInfo” Class to “Course”:

Originally, the abstract class that represents the generalised information about a course that stays consistent year-after-year was stored in a class called “CourseInfo”. We have decided to change the name from “CourseInfo” to “Course” because it is a more accurate name in the context of our design. The “CourseOffering” Class will inherit this abstract “Course” class. Since inheritance is a “Is a” relationship, it makes much more sense to say that “a CourseOffering is a Course” instead of saying that “a CourseOffering is a CourseInfo”.

Renamed “Profile” Class to “AcademicAssociate”:

Originally, this class was named “Profile” because it represents the generalised object that the user uses to interact with the website. However, we have decided to change the name of this class from “Profile” to “AcademicAssociate” because it works much better in the context of our design domain and the relationships that we specify on our class diagrams. Originally, the “Student” and “Professor” classes inherit the “Profile” class, which implies that “a Student is a Profile” and “a Professor is a Profile”, which are both inaccurate statements. By changing the name from “Profile” to “AcademicAssociate”, our inheritance relationships make more sense.

Removed course section maximum capacities for registration:

Originally, each individual CourseOffering instance contained a mapping from a section to the maximum registered student capacity for that specific section. For ease of implementation and development speed, we have decided to remove this functionality from our demo and allow a student to register in any available section. Of course when this software is implemented for real life use, this functionality will be implemented using real data stored in backend databases.

Added “courseCredits” attribute to the “Course” Class:

Every Course instance should contain an attribute for the amount of credits received for completing the course. This attribute should have been added in the classlist from the beginning but we accidentally forgot to include it. Our implementation needs this attribute for display purposes, so we have fixed our mistake by adding a “courseCredits” attribute to the “Course” class that represents the amount of credits received for completing the course.

Removed the “maxCreditsForCurrentSemester” attribute from “Student” Class:

Originally, every Student instance contained a “maxCreditsForCurrentSemester” that prevented the student from registering for more than 2.5 credits in a single semester. For ease of implementation for demo purposes, we have decided to remove this functionality and allow the user to register for as many courses as they want.

Implemented Return Codes after completing Course Scheduling Actions:

Originally, each user action function was implemented to print to the terminal once the action was completed. This is a very disorganised and procedural mindset, so we have instead opted to return status codes to the Main class, who can then display the appropriate messages as needed.

Removed many Getters/Setters from various classes:

Originally, we wanted each class to have getter and setter functions for each individual attribute, but we decided to remove many of these functions in our implementation because they expose our data and should not be used in Object-Oriented Design very often anyway. Instead, we implemented functions that only return results after the task is completed instead of data that has to be processed further.

Prompt user for names on login:

Originally, our program would only ask if the user is a professor or a student. Now, our program also prompts the user for their name on login, as we have realised that much of our functionality cannot be implemented without having the name of the user. We also can improve the user experience by displaying their name on the welcome screen.

Implemented functionality for editing the courseOffering object when registering or dropping a courseAttempt:

In our original design, we didn’t implement functionality for updating the arrayLists in the courseOffering instance when its corresponding courseAttempt was registered or dropped by the user. As a result, we had to add functionality to update the courseOffering arrayLists when a courseAttempt is registered or dropped.