Lab 7:  
ELEC 3225

Anthony Huynh, Inderdeep Singh, Sultan Jan  
August 2nd, 2022

For the Summer 2022 semester, a team consisting of Sultan Jan, Inderdeep Singh, and Anthony Huynh has been assigned to design a university’s scheduling system. The assignment had a list of required features that were expected to be implemented. First, the system had to be compatible with three distinct types of users. Beginning with the student users, students should be able to add and remove courses from their own schedule based on a course’s specific ID number. Additionally, the system should be able to detect if there are any conflicts or overlapping courses in the student’s schedule. Finally, the student should be able to print out their schedule. Moving to the next type of user, the instructor users should be able to print their course teaching schedule, print out the system’s course roster, and search for courses based on specific parameters. Lastly, the admin users will be able to add or remove courses from the system, add instructor or student users, and add an instructor or a student to a distinct course. Furthermore, All users should be able to log in and out of the system freely, print out a list of all courses within the system, as well as search for specific courses based on certain parameters.

Using process and UML models we’ve created as a guide, we have designed, implemented, and tested our code numerous times to ensure the program meets the requirements. Starting with the process models, the three of us each created a “waterfall”, “incremental”, and an “integrate and configure” model. After much deliberation, we chose to implement Anthony’s incremental model (Fig. 1) for the rest of the duration of this project.

Next, activity models, use-case diagrams, class diagrams, and sequence diagrams were created to help further guide our development. Inderdeep Singh created the activity models which consisted of one for adding and removing courses from the semester schedule (Fig. 2), one for assembling and printing a course roster for the instructor class (Fig. 3), and one for adding and removing courses from the system as an admin user (Fig. 4). In addition, Inderdeep also created the use case diagrams for the base user (Fig. 5), student (Fig. 6), instructor (Fig. 7), and admin classes (Fig. 8). Sultan Jan created the class diagram (Fig. 9). Anthony Huynh created the sequence diagrams for logging in and out (Fig. 10), searching through all available courses (Fig. 11), and searching for specific courses based on distinct parameters (Fig. 12).

Overall, the program is successful and fully functional, but it can still be improved upon. If we had to evaluate our code’s efficiency, it would be rated 84% efficient. There are some sections of code that may be more complex than necessary. Although it works as intended, an example of the least efficient section of code is the student or instructor's adding courses to a schedule function. The way the code is currently written, the program first checks the student or instructor table within the user database. Then, it takes the string of all assigned courses from the user’s “courses” column and stores it as a variable. The string is then parsed into individual strings so that the individual courses can be checked one by one to see if there are any scheduling conflicts. If there are no conflicts, the course is added to the user’s schedule. This is the least efficient portion since it requires significantly more resources and time compared to the other functions. This could be improved by decreasing the complexity and steps needed within the code. The most efficient portion of the program would be all the admin user’s functions. Adding courses to the system is as simple as inputting all the required information. The information is then added to the courses table as its own course. Similarly, adding instructor or student users will also take the direct input and create users using the information.

|  |
| --- |
|  |
| (Fig. 1) Incremental Model. |

|  |
| --- |
|  |
| (Fig. 2) Activity model for adding and removing courses from a schedule. |
|  |
| (Fig. 3) Activity model for assembling and printing a course roster. |
|  |
| (Fig. 4) Activity Model for Adding and removing courses from the system as an admin class. |

|  |  |
| --- | --- |
|  |  |
| (Fig. 5) Use case diagram for the base user class. | (Fig. 6) Use case diagram for the student class. |
|  |  |
| (Fig. 7) Use case diagram for the instructor class. | (Fig. 8) Use case diagram for the admin class. |
|  | |
| (Fig. 9) Class diagram for the four user types. | |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| (Fig. 10) Sequence diagram for logging in and out. | (Fig. 11) Sequence diagram for searching all courses. | (Fig. 12) Sequence diagram for searching through all courses based on specific parameters. |

Final Repository: [Github](https://github.com/anthonyhuynh729/Course-Registration)

Video: [Applied Programming Course Registration Project.mp4](https://mywentworth-my.sharepoint.com/:v:/g/personal/huynha3_wit_edu/ETHPeFxsg7pCi083GSaimp0BAo6HyfpjS6EQqk9tjdEbmw?e=CLvI56)