**SEG2105A Assignment 5**

1. *The Problem*

University of Ottawa students have no easy way to generate a conflict-free, optimised schedule.

2. *System Description*

The system will be used by students looking to generate an optimized timetable. A student will specify which semester they are planning for, and what courses they want to take. Courses are made up of sections. Each section will contain at least one activity. Activities can be lectures, seminars, discussion groups, tutorials, and laboratories.  More activities are optional. The section will specify the number of lectures, labs, and tutorials required. If a minimum has not been specified of any one kind of activity, then all activities of that type are required. The student should be able to specify when he or she does not want to be in a certain section, due to, for example, an unwanted professor or that section being full. Once all course and section selections have been made, system will generate all possible conflict-free schedules.

Students will be able to sort the schedules with options prioritizing number of days off, longest breaks, shortest days, latest start, or earliest finish. The student can also choose a set of courses, and from this set choose a minimum number that will be placed on the schedule. For example, if a student must choose two electives out of four possible elective courses, the generated schedules will contain **every** combination of two out of the four electives, and sorting options will help students choose the best. The student should be able to print the schedule, including a color-coded week-long calendar and a table that summarizes the course-code information for quick reference when registering. The student should also be able to save the information to a file, along with the option of opening another configuration or starting a new one.

3. *Some user stories:*

**Student:** I want to make a schedule that is conflict-free and doesn’t have many evening classes so that I can sleep more.

**Student:** My friends are taking the same courses as me. I will save my configuration and send them the file to save them some time.

**Student:** I don’t like a certain Professor. I’m going to exclude all the sections with that professor because I want a teacher that has a better reputation.

**Student:** I want to drop a course, so I’m going to see if dropping this course allows me to have a day off. The easiest way to do this is to edit my configuration file and remove this course, then prioritize “day off” in sorting.

**Student:** I have one elective to take, and three choices. I want the one that lets me wake up the latest. Using the scheduler, I’m going to put these courses in a set next to with my mandatory courses and see which one fits my schedule best, using sorting options.

4.       Architecture

The system uses a client-server architecture. The coding language will be Java, using OCSF. The server will be controlled via console, including only simple actions such as restart, stop, quit, reload, etc. It will also display where connections are coming from at any given time.

Upon startup, the server will load in a comma-separated file that holds all course information for the current semesters. The information will be held in a list, possibly an array list, of “Course” objects. Each Course will have a description and a list of Sessions. Sessions have an ID (course code + a letter), a semester, and a collection of Activities. Each Activity has a type (lecture, seminar, discussion group, tutorial, laboratory), a number (e.g.: lecture 2), a day of the week, a start time, an end time, a location, and the name of the professor.

After starting up and loading the database of courses into memory, the server starts listening for clients. The client automatically connects at startup, having an option to change the defautl server address using a settings pane. The entire interface for the client is graphical. After connecting to the server, the client may start adding courses through an interface. First, the client selects a

5.       UML

6.       Messages