SYSC 4504 Project : Courses Selection Assistant

Application objective: Students select their courses for each term, based on their program requirements, and based on the course availability and schedule.

Example **:** Communications Eng. take in the Winter of Year 2 (WY2) :

- MATH 3705 (4 lectures sessions and 4 lab sessions for each lecture session; students must register in the lab sessions with respect their lecture session, ie. Students in A can select only lab A1, A2, A3 or A4.)

- SYSC 2003 (1 lecture session and 3 lab sessions)

- SYSC 2004 (2 lecture sessions and 4 lab sessions)

- ELEC 2507 (2 lecture sessions and 6 lab sessions)

- ELEC 2607 (3 lecture sessions and 14 lab sessions)

- One Basic Science Elective

The objective of the application are :

(i) identify, and display the courses (Lec and Lab) that a student can take and register based on the courses (s)he completed previously and the course status,

(ii) maintain an up-to-date courses database (A student who uses this application in june will have more options than a student who uses it in late august, as many courses are already full. The courses list must maintain an up-to-date number of seats available.)

Application specifications:

- Specify the list of courses to take in each term: FY1 (Fall Year 1), WY1 (Winter Year 1), FY2, WY2 … FY4 and WY4. This list must include the basic science electives, the complementary studies electives and the breadth electives and general engineering electives. See prerequisite trees of all programs at :

http://www.carleton.ca/engineering-design/current-students/undergrad-academic-support/prerequisites/

- Store the courses prerequisites. For instance, to take MATH 3705, a student must complete MATH 2004 AND MATH 1005. To take SYSC 4907, a student must take concurrently ECOR 4995, and complete all 1st, 2nd year courses and at least 3.5 credits of 3rd year (the completed credits do not include CCDP 2100 and complementary studies

electives). To take SYSC 3303, a student must complete SYSC 2003 AND (SYSC 2004 OR SYSC 2100).

- Store the courses offering. Each course is characterized by its lectures (term (F or W) , day, time, session, current capacity) and its lab (term, day, time, session, current capacity), and the relationship between LEC and LAB. For instance, a course with LEC A and B, and LAB L1, L2, L3 allows the students to register in either A or B and pick arbitrarily a LAB session. On the other hand, a course with LEC A and B, and LAB A1, A2, B1, B2 allows the students to register in either A or B but imposes the students to register in either A1 or A2 when (s)he registers in LEC A.

- Store the list of electives :

Basic Science Electives: http://www.carleton.ca/engineering-design/current-students/undergrad-academic-support/engineering-design/students/academic-support/basic-science-elective

Complementary Studies Electives : http://www.carleton.ca/engineering-design/current-students/undergrad-academic-support/engineering-design/students/academic-support/cse

Communications Eng. Electives, Biomed. Electives, Software Eng. Electives, General Engineering Electives:
http://calendar.carleton.ca/undergrad/undergradprograms/engineering/

- Get the student's input. Each student using the application is requested (if necessary) to provide the courses (s)he completed already. The application must offer the 2 options: "I am on-pattern" and "I am off-pattern". If off-pattern, the student will specify all completed courses. If on-pattern, the student can specify only if (s)he completed 1st, 2nd or 3rd year courses.

- Identify the list of courses a student can take based on student's input and based on the courses availability and prerequisites. There may be courses that the student can take, but they may be already full at the time the student uses the software.

- Get the student's confirmation that (s)he accepts the suggested list and update the courses status. One seat is now assigned to a student and one seat must be removed from the courses database.

Project development tasks:

The project must be conducted in groups of two or three students. The software you will develop can be used by any student in particular students from one of the following programs: Biomedical and Electrical Eng., Communications Eng., Computer Systems Eng., Software Eng.

Task 1 : How to store the list of courses to take in each term and in each year and for each of the 4 programs ? How to store the prerequisites of a course ? Note that there are various types of prerequisites. Refer to the examples presented previously.

Task 1 does not require any implementation. It only requires that you define the data structures (or database table structure) that you will use and implement. You may use Database, and/or Files to store the data.

Task 2 : How to store the courses offerings (term, LEC, LAB, capacity) ?

Task 2 does not require any implementation. It only requires that you define the data structures (or database table structure) that you will use and implement. You may use Database, and/or Files to store the data.

Task 3 : How to store and access/update the list of electives (the project must consider all types of electives) ?

Task 3 does not require any implementation. It only requires that you define the data structures (or database table structure) that you will use and implement. You may use Database, and/or Files to store the data.

Task 4 : Design and implement the student View 1 which obtains from the student all the relevant information (program, on/off pattern, list of completed courses).

Two versions are required : one in Java and one in HTML (and javascript).

Task 5 : Design and implement the server process : From the student input, the server process executes the following tasks:

- Step 1 : Determine the list of courses the student can take based on the courses already completed.

- Step 2 : Determine which courses are available this coming term and which still has some available seats. If the student uses this application in June-September, the server process will identify the courses available for the fall. If the student uses the application in December-January, the server process will identify the courses available for the winter term. The application will focus on one term at a time.

- Step 3 : Build the timetable of the student, only for courses which are required by the program. For the electives, identify the courses the student can take, based on prerequisites, availability and schedule.

The server process must be implemented in PHP.

Task 6 : Implement Student View 2: Present the results to the student. This View asks the student to select the electives (s)he wants to take and to confirm that (s)he will register in the proposed courses.

Task 7 : From the student's confirmation, update the database of courses change the number of available seats. Note that during Task 6 and Task 7, some other students may use the application and register in some courses (if it happens, the capacity of the course was updated) !

Marking Scheme :

Task 1 (3)	3			
Task 2 (6)	3			
Task 3 (9)	3			
Task 4 (20)	**Rubrics for Task 4 (for Java and HTML/javascript)**			
	Basic (2)	**A little better (5)**	**Good Project (8)**	**Professional (11)**
	Student View 1 is implemented in basic HTML (eg: Student on pattern, and the application still shows and asks for all completed courses).	Student View 1 asks only relevant information but does not include any validation. (eg. Student enters SYSS 3303; the server finds that it is not a valid course, while the validation should be done in Client view.)	Student View 1 requests only appropriate information, includes student's inputs validation (and requests new valid inputs when inputs was invalid)	Student View 1 is a "good project" and looks like a professional work (layout, color scheme, dynamic features)
Task 5 (20 + 15 + 25 + 15 = 75)	**Rubrics for Task 5 (implementation of the courses selection based on prereq - Step 1)**			
	Insufficient (0)	**Developing (5)**	**Accomplished (10)**	**Exemplary (15)**
	The program does not take into account courses prereq.	Only the simple prereq (ie: Course 1 is required to take Course 2) is implemented	More complex prereq (such as course 1 AND (Course 2 or Course 3) are required to take Course 4)	All prereq including year status in Eng are implemented correctly.
	Rubrics for Task 5 (implementation of Step 2 : list of courses selection based on courses availability (term and capacity)			
	Insufficient (5)	**Developing (10)**	**Accomplished (15)**	**Exemplary (25)**
	The program takes into account only the course capacity	The program takes also into account lec/lab but the algorithm used to identify the course is not fully correct or complete	The program takes into account capacity, lec/lab and the algorithm selecting the course is correct and complete	The course selection works correctly and produces at least two schedules (when possible) to let the student choose. The two schedules may include the same set of courses, but have

				scheduled on different days or at different in the day
	Rubrics for Task 5 (implementation of the results to send to the student)			
	Insufficient (0)	**Developing (5)**	**Accomplished (10)**	**Exemplary (15)**
	Incomplete or incorrect schedule of courses, with or without the electives	The result includes only one schedule with the required courses only	The result has only required courses and omits the electives, and includes at least two possible schedules	The result includes at least two possible schedules and the list of all valid electives
Task 6 (75 + 15 = 90)	**Rubrics for Task 6 (presenting the results and getting new inputs from student)**			
	Insufficient (0)	**Developing (5)**	**Accomplished (10)**	**Exemplary (15)**
	Basic text/html to display the results and to get the confirmation from student	The results are presented in tables, the electives are presented as plain text. The program does have confirmation and electives selection	The results are presented in tables, with the electives as list or combo box (along with their schedule). The program does have confirmation and electives selection.	Whenever the student selects an elective, the program checks whether the course is still available. (see "Exemplary" description for next Task)
Task 7 (100)	**Rubrics for Task 7**			
	Insuficient (0)	**Developing (4)**	**Accomplished (7)**	**Exemplary (10)**
	Courses availability update is not implemented	Incomplete update of courses availability (eg. Lec is updated but not lab)	The state of the courses is fully updated.	The state of the courses is updated. There is a class that checks in real-time the course availability and processes requests from Task 6.
Deployment (115)	**Rubrics for deployment**			
	Insufficient (0)	**Developing (5)**	**Accomplished (10)**	**Exemplary (15)**
	You would not even use your	The app is written only for	User manual is provided and	User manual is provided and

	own app.	one program, as all inputs (prereq, list of electives ...) are hard-coded in the classes.	contains files format or data structures when providing list of electives, prereq trees. When deploying this software, we only need to provide those input files for all programs.	contains files format or data structures when providing list of electives, prereq trees. When deploying this software, we only need to provide those input files for all programs. The software also includes an admin view to input and update the input files
Students evaluation (125)	Each student must evaluate the app written by the other groups work and assign a grade of 0 to 10. The average of all grades will be added to the project grade.			
Quality of the code (140)	**Rubrics for quality of the code**			
	Insufficient (0)	**Developing (5)**	**Accomplished (10)**	**Exemplary (15)**
	The entire app is written in few files. The code has no particular structure no object oriented approach. Very poor on non-existent documentation.	The project does not use OO programming. The files/ classes are based only on functions. Though the code is fairly well explained, the files could have been stored in folders based on their functionalities.	All the files are stored in dedicated folders with explicit files names. The project uses OO programming efficiently.	All the files are stored in dedicated folders with explicit files names. The project uses OO programming efficiently. If addition or extension is necessary, one could easily find which file/class to modify.

Your work will be evaluated using XAMPP as web server. Your work **cannot integrate or use** any additional software package or library (such as codeigniter, cakephp, jquery ...).

Suggested Timeline :

Task 4 (Student View 1)

Version 1 (Basic html) : mid september.

Version 2 (Dynamic display of components with javascript) : mid october

Task 5 / 7(Server implementation)

Version 1 (Getting input from student) : end september

Version 2 (use of files and database) : october

Task 6 (display results to student)

Version 1 (basic html table) : mid september

Version 2 (validating inputs) : end of october

Instructions for project submission:

Submission of the project before November 28, 16:00.

Zip all the classes, and files you wrote for this project.

Along with your program :

(1) Add a brief report providing the contribution of each project member (who was the leader of each task or step, what is the exact contribution in the coding of each files/class)

(2) You may create folders in your project (eg: client/html/ client/java/ server/, server/db, server/resources/ ...) In each folder, create a readme.txt file to provide the content and the purpose of the folder. For each class/file in the folder, state briefly the purpose of the class/file (eg : php class to get inputs from user and validate their correctness; or php class containing all operations to access database ...) in the readme.txt file. If you have all your files in one unique folder, do have the readme.txt file.

(3) Provide the instructions on how to install your program (what needs to be setup ? Database ? Table structures ? File systems ?) in a php script named install.php.

This document may be updated to include additional instructions or help. Announcement will be made in class when updates are made.