Concordia University

Department of Computer Science

and Software Engineering

Software Process

SOEN 341/4 --- Winter 2016 --- Section S

Deliverable 1

February 10th, 2016

Project Name: Apollo

Team Name: Athena

Team Members Information	
Name	**SID**
Philippe Abou Kasm	27305133
Wahab Ahmed	21311980
Sabrina Ashraff	29533400
Francis Bouchard	26786812
Clozzy-Mailey Chavez	26610315
Ricardo Cortés	27734107
Liuai Hatter	25976618
Jian Huang	26772862
Anna Rogozin	27494939
Ramy Sandouk	
Matthew Teolis	40005332

Table of Contents

1.Grading Sheet

Section	Evaluation criteria (see instructions in the template for details)	Grading
all	10 marks are allocated for excellence, professionalism and quality of work above and beyond the correct meeting of specifications..	/10
1	Presentation of this document	/5
2	Completeness and accuracy with regard to initial project description	/1
3.1 . . 3.2 3.3	Completeness and accuracy of the project functional requirements expressed as formal use cases, including difficulty and importance indicators completeness and accuracy of the diagram and description of the domain model completeness and accuracy with regard to initial project description accuracy with regard to initial project description, difficulty and importance ratings	/15 . /3 . /1
4.1	Description of all team members' capacities and schedule restrictions	/1
5	List of goals removed from the project. For each goal removed, give justifications in light of the resources available	/`1
6.1 . 6.2	Clarity of textual description, validity of rationale, clarity and appropriateness of diagram, list of modules responsibilities List of technologies used, validity of rationale	/2 . /1
7.1 . 7.2 . 7.3 . 7.4 7.5 7.6	Completeness of list of activities, clarity of their stated purpose, as well as statement of what artifacts they are producing Completeness of list of artifacts to be produced during the project, validity of roles description of each artifact Cost estimation of each individual artifact, validity of explanation of cost estimation, total cost estimate Mapping of activities to individual project members Accurate and complete presentation of milestones Assessment of risks `	/1 . /2 . /2 . /1 /1 /1
8	Early Prototyping	/2
Total		/50

DO NOT REMOVE THIS PAGE WHEN SUBMITTING YOUR DOCUMENT

2. PROJECT DESCRIPTION

The purpose of this system is for a student enrolled in the Software Engineering program to plan their schedule for the entire duration of the program.

The application will have the following class of users:

- Students
- Professor
- Faculty Administrator
- System Administrator

With this application, the student will also be able to define their program option in Software Engineering from amongst the following four options:

- General
- Computer Games
- Web Applications
- Real-Time and Embedded Systems

Once the student has defined their program option, a master course list will be provided with all the courses the student is required to take. From there, a student will be able to select the courses they wish to take each semester for the duration of the program.

Several settings for preferences will be featured, such as:

- Creating the course schedule based on the course sequence provided by the department for each option
- Creating the course schedule without depending on the suggested sequence
- Selecting time preferences for courses, either morning, afternoon, or night

Once all these preferences are selected by the student, the Apollo application will evaluate the student record based on the following criteria:

- The overall academic record of the student
- Checking if each course's prerequisites have been met
- Checking if the credit requirements are present
- Courses for which the student has an exemption
- Assuring if the courses are taught by an accredited engineer

Once all these checks have been made, Apollo will generate the number of possible schedules that the student can choose from, and the student can then select the schedule they prefer.

3. GOALS AND CONSTRAINTS

In this section, all the features that the Apollo application will implement will be discussed, as well as the goals which will be achieved. This will be explained by presenting the functional requirements of the application in terms of the users which will be expected to use it, and use case models to explain each feature. A domain model diagram is provided to give an overall view of how the various entities within the software will interact. Finally, constraints which will be anticipated in the building of this application are discussed.

3.1 Functional Requirements

1. **User**
   - 1.1. All users can log in
   - 1.2. All users can log out
   - 1.3. All users can update their personal information

2. **System Administrator**
   - 2.1. System administrator can create new users
     - 2.1.1. System administrator can grant new users specific permission
   - 2.2. System administrator can view existing users
   - 2.3. System administrator can update existing users' information
     - 2.3.1. System administrator can change users' permissions
   - 2.4. System administrator can remove existing users from the system.

3. **Faculty Administrator**
   - 3.1. Faculty administrator can create new courses
     - 3.1.1. Faculty administrator can assign requisites and requirements to courses
   - 3.2. Faculty administrator can view existing courses
   - 3.3. Faculty administrator can update existing courses' information
     - 3.3.1. Faculty administrator can change course requisites and requirements
   - 3.4. Faculty administrator can remove existing courses from the system

4. **Professor**
   - 4.1. Professor can view his or her course schedule
   - 4.2. Professor can update students' grades

5. **Student**
    5.1. Student can view his or her current schedule
    5.2. Student can review his or her course history
    5.3. Student can search for courses
        5.3.1. Student can view a course's description
    5.4. Student can generate schedules
        5.4.1. Student can choose course preferences to generate schedules
        5.4.2. Student can save generated schedules
        5.4.3. Student can view saved schedules
        5.4.4. Student can finalize a schedule
    5.5. Student can print a schedule

Use Case Descriptions

UC01 - Login	
Risk Assessment	5/5
Importance	5/5
Actor(s)	Student, Professor, Faculty Admin, System Admin
Description	Validate user credentials and allow user specific privileges to the system.
Basic Flow	1. User visits scheduler site 2. User enters credentials 3. Website redirects user to main page
Pre-condition	User is not logged in.
Post-condition	Success: User is logged in. Fail: No access granted. User is not logged in and remains in login page.

UC02 - Logout	
Risk Assessment	5/5
Importance	3/5
Actors	Student, Professor, Faculty Admin, System Admin
Description	Terminate the session of the user interacting with the system, no longer provide the user his or her personal data.
Basic Flow	1. User logs out 2. Website redirects user to login page
Pre-condition	User is logged in.
Post-condition	Success: User is logged out. Fail: User is still logged in.

UC03 - Update Personal Information	
Risk Assessment	5/5
Importance	5/5
Actors	Student, Professor, Faculty Admin, System Admin
Description	Update user profile information.
Basic Flow	1. User fills in personal information in specified fields 2. User saves the newly entered information
Pre-condition	User is viewing personal information page.
Post-condition	Success: User's information is changed. Fail: User's information is not changed.

UC04 - Review Course History	
Risk Assessment	2/5
Importance	4/5
Actors	Student
Description	Review student's course history which includes the semesters taken and student's grades.
Basic Flow	1. User requests to view course history 2. System redirects user to course history page
Pre-condition	User is logged in.
Post-condition	Success: User sees course history. Failure: User does not see course history.

UC05 - View Degree Audit	
Risk Assessment	2/5
Importance	2/5
Actors	Student
Description	Review list of completed courses and remaining courses to be taken in order to graduate.
Basic Flow	1. User requests to view degree audit 2. System redirects student to degree audit page
Pre-condition	User is logged in.
Post-condition	Success: User sees course audit list. Fail: User does not see course audit list.

UC06 - Choose Course Preferences	
Risk Assessment	2/5
Importance	3/5
Actors	Student
Description	Edit preferences, add constraints such as class time preference and semester preferences.
Basic Flow	1. User chooses day, time, or semester preferences 2. System saves user's preferences
Pre-condition	User is logged in.
Post-condition	Success: User preferences are saved. Fail: User preferences are not saved.

UC07 - Search Courses	
Risk Assessment	1/5
Importance	5/5
Actors	Student, Faculty Admin
Description	Search for a specific course by course name or by department.
Basic Flow	1. User enters search field information 2. User requests to view courses matching query 3. System redirects user to found courses page
Pre-condition	User is logged in.
Post-condition	Success: Find course matching to query. Fail: No course found that matches query.

UC08 - View Course Description	
Risk Assessment	1/5
Importance	4/5
Actors	Student, Faculty Admin
Description	See course instructor, sections, session time, requisites.
Basic Flow	1. User selects the course he or she would like to view 2. System displays the selected course's description
Pre-condition	User is logged in and course is existent.
Post-condition	Success: Course information is displayed. Fail: Course information is not displayed.

UC09 - Generate Schedules	
Risk Assessment	1/5
Importance	5/5
Actors	Student
Description	User can view the different semesterly schedules offered to him or her.
Basic Flow	1. User chooses course preferences 2. User requests to generate schedules 3. System displays semester schedule
Pre-condition	User is logged in and has chosen the time frame for which to display schedules and entered his/her preferences.
Post-condition	Success: One or more schedules are generated according to user's specifications. Fail: No schedules are generated.

UC10 - Generate Sequences	
Risk Assessment	1/5
Importance	5/5
Actors	Student
Description	User can view the different course sequences offered for their remaining semesters until graduation.
Basic Flow	1. User chooses course preferences 2. User requests to generates sequences 3. System displays sequences
Pre-condition	User is logged in and has entered their preferences.
Post-condition	Success: One or more sequences are generated according to user's specifications. Fail: No sequences are generated.

UC11 - Save schedule	
Risk Assessment	1/5
Importance	4/5
Actors	Student
Description	User can save generated schedule.
Basic Flow	1. User selects schedule to save 2. Schedule is saved in user's profile
Pre-condition	User is logged in and sees at least one generated schedule.
Post-condition	Success: Schedule is saved to user profile. Fail: Schedule is not saved to user's profile.

UC12 - View Saved Schedules	
Risk Assessment	1/5
Importance	3/5
Actors	Student
Description	Student views previously saved schedules or sequences that were generated from scheduler.
Basic Flow	1. Student runs scheduler 2. Student saves schedule(s) 3. Student requests to view their saved schedules 4. System displays student's saved schedules
Pre-condition	Student is logged in.
Post-condition	Success: Student sees saved schedules. Fail: Student does not see saved schedule page.

UC13 - View Student Schedule	
Risk Assessment	1/5
Importance	5/5
Actors	Student
Description	Student sees their current semester schedule.
Basic Flow	1. Student navigates to their current semester schedule. 2. System displays current schedule.
Pre-condition	Student is logged in.
Post-condition	Success: Student sees current schedule. Fail: Student does not see a current schedule.

UC14 - Print Schedule	
Risk Assessment	1/5
Importance	1/5
Actors	Student
Description	User prints their current schedule, generated schedule, or saved schedule.
Basic Flow	1. User requests to print his/her schedule 2. System displays schedule print page
Pre-condition	User is logged in and has finalized a schedule.
Post-condition	Success: User's schedule is printed. Fail: User's schedule is not printed.

UC15 - Create Course	
Risk Assessment	3/5
Importance	5/5
Actors	Faculty Admin
Description	Faculty Admin creates a new course.
Basic Flow	1. User navigates to create course page 2. User enters all course information 3. User saves new course
Pre-condition	Faculty Admin is logged in and has permission to add course.
Post-condition	Success: Faculty Admin creates a course in the system. Fail: Course is not added or Faculty Admin is restricted from adding courses.

UC16 - Remove Course	
Risk Assessment	5/5
Importance	4/5
Actors	Faculty Admin
Description	Faculty Admin removes a course.
Basic Flow	1. User searches a course 2. User finds course to remove 3. User removes the course
Pre-condition	Faculty Admin is logged in and has permission to remove course.
Post-condition	Success: Course is removed. Fail: Course is not removed or Faculty Admin is restricted.

UC17 - Update Course	
Risk Assessment	2/5
Importance	5/5
Actors	Faculty Admin
Description	User edits the description of a course.
Basic Flow	1. User enters information for a course 2. System updates information for that course
Pre-condition	User is logged in.
Post-condition	Success: The specified course's information is updated. Fail: The course's information is not updated.

UC18 - Create User	
Risk Assessment	4/5
Importance	5/5
Actors	System Admin
Description	System Admin creates a user of the system.
Basic Flow	1. System Admin enters information for a new user (with specific permissions) 2. System creates the new user
Pre-condition	System Admin is logged in.
Post-condition	Success: A new user is created. Fail: A user is not created.

UC19 - Remove User	
Risk Assessment	4/5
Importance	5/5
Actors	System Admin
Description	System Admin removes a user from the system.
Basic Flow	1. System Admin selects a user to remove 2. System removes selected user
Pre-condition	System Admin is logged in.
Post-condition	Success: The user is removed from the system. Fail: The user is not removed from the system.

UC20 - Update User	
Risk Assessment	4/5
Importance	5/5
Actors	System Admin
Description	System Admin updates a user's information.
Basic Flow	1. System Admin modifies information of a selected user 2. System updates the user's information
Pre-condition	System Admin is logged in.
Post-condition	Success: The user information is updated. Fail: The user information is not updated.

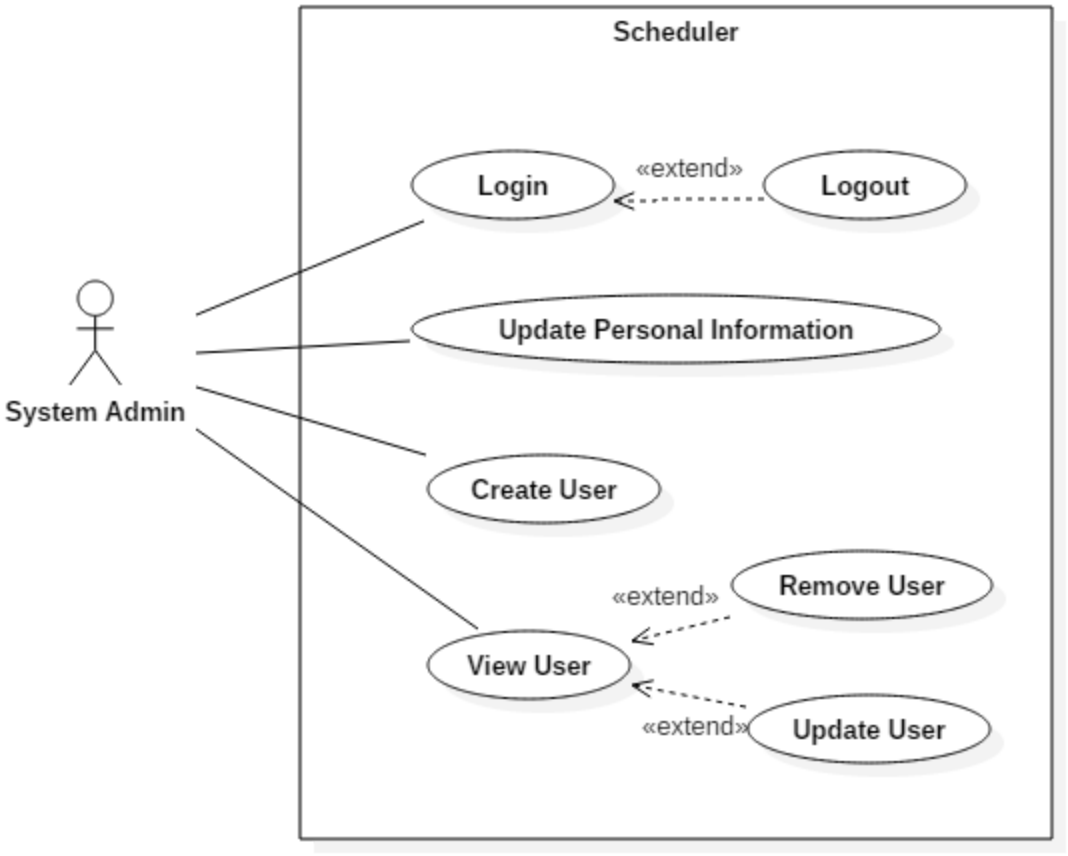
UC21 - View User	
Risk Assessment	1/5
Importance	4/5
Actors	System Admin
Description	View system user information.
Basic Flow	1. System Admin logs in 2. System Admin searches for a user 3. System Admin views user information
Pre-condition	System Admin is logged in and finds user.
Post-condition	Success: System Admin sees user information. Fail: User information is not displayed.

UC22 - Update Course Grades	
Risk Assessment	4/5
Importance	5/5
Actors	Professor
Description	User enters the grades of all students of a class.
Basic Flow	1. User enters student grades for a course 2. System updates user grades for that course
Pre-condition	User is logged in and is the professor of the selected course.
Post-condition	Success: Students' grades in the selected class are updated. Fail: Students' grades are not updated.

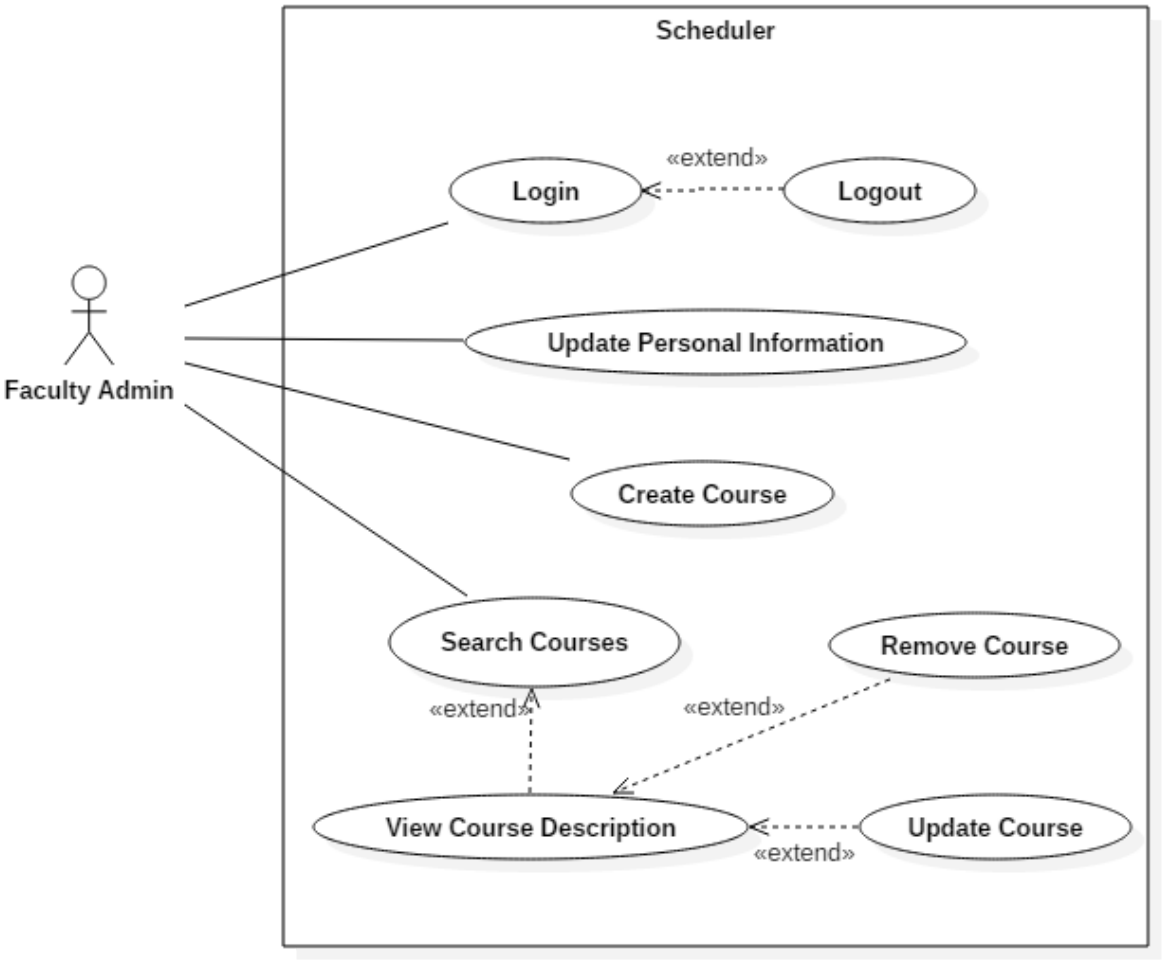
UC23 - View Professor Schedule	
Risk Assessment	1/5
Importance	3/5
Actors	Professor
Description	Professor views the schedule of the course they are teaching.
Basic Flow	1. User navigates to view schedule page 2. System displays professors schedule
Pre-condition	User is logged in and is a professor.
Post-condition	Success: Professor sees their current schedule. Fail: Schedule is not displayed.

Use Case-Models

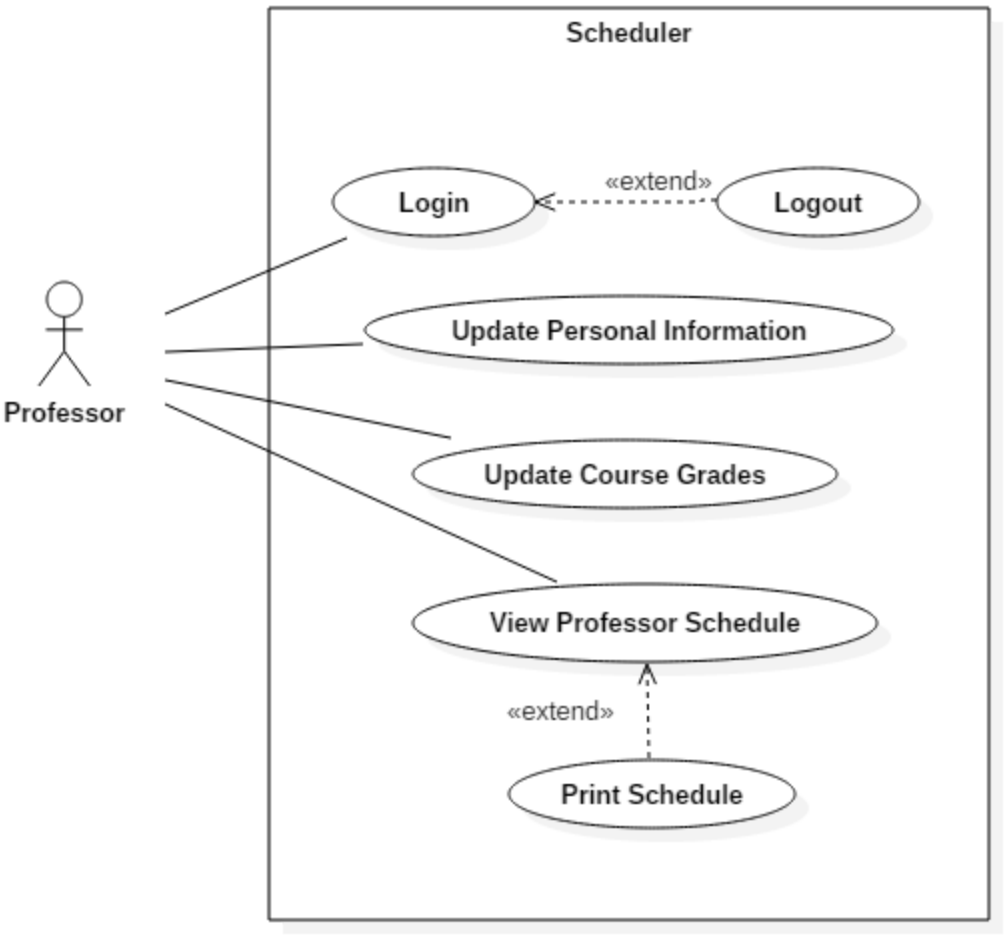
System Administrator



Faculty Administrator



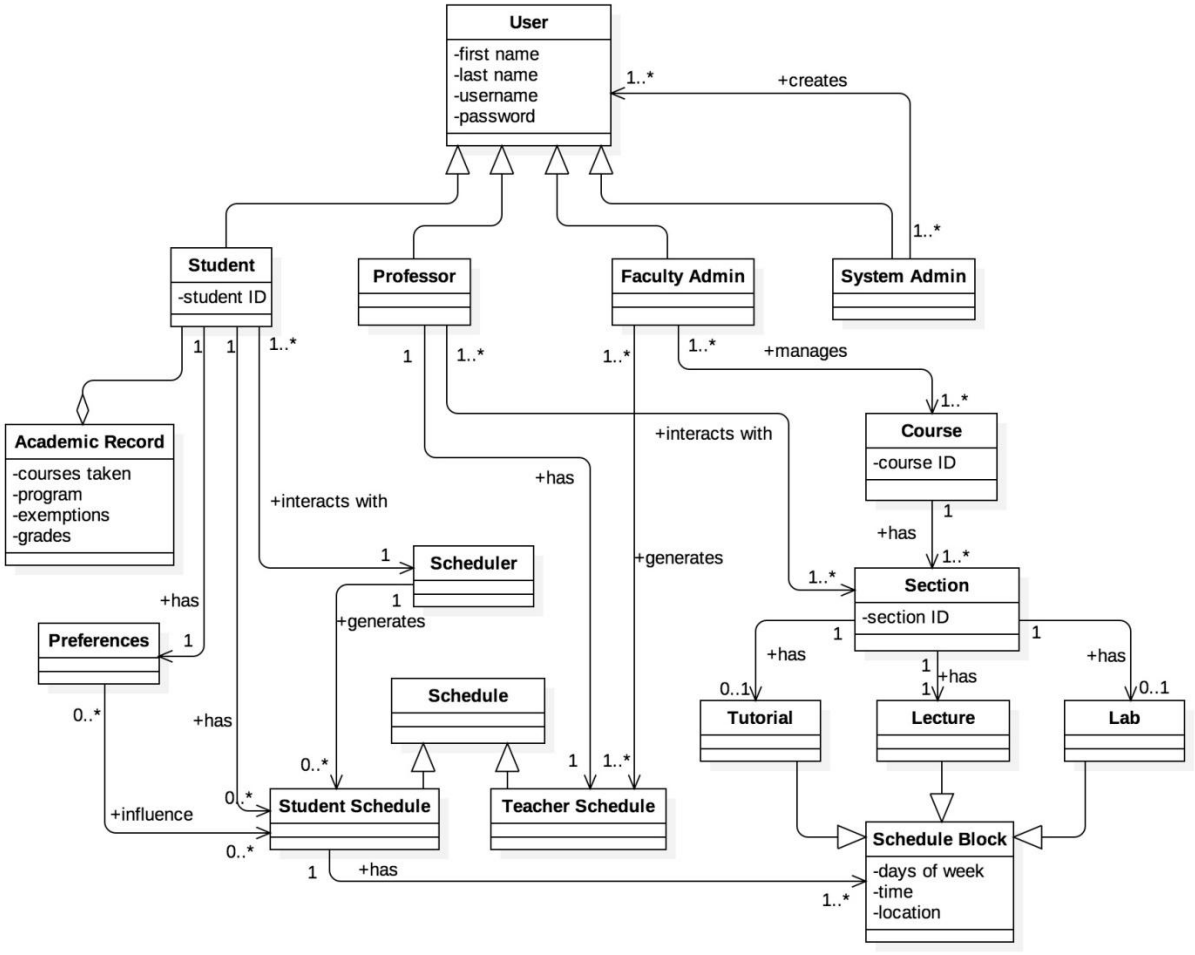
Professor


Scheduler
Login
«extend»
Logout
Update Personal Information
Professor
Update Course Grades
View Professor Schedule
«extend»
Print Schedule

Student


Scheduler
Login
«extend»
Logout
Update Personal Information
Review Course History
View Degree Audit
Search Courses
View Course Description
«extend»
Student
Choose Course Preferences
«include»
«include»
Generate Schedules
Generate Sequences
«extend»
«extend»
Save Schedule
«extend»
View Save Schedules
View Student Schedule
Print Schedule
«extend»

3.2 Domain Model



3.3 Constraints and Quality

Based on ISO/IEC 9126 [1] standard, the following quality criteria should been address in the design:

1. **Functionality**
   The constraints described in this section address features that extend functionality to technical aspects.
   a. Security
      i. The access to the system will be granted only by providing correct credentials (i.e. username and password)
      ii. The password must have a length between 8 to 16 characters alphanumeric characters.
      iii. The communication protocol must be encrypted (i.e. the system will be access by HTTPS)

iv. iv. The user's password will be stored encrypted in the database, using SHA-2
v. After 10 minutes of inactivity, the system shall automatically logout the user.
vi. The system must provide the options to manage users and permissions.

b. Interoperability
   i. The system must have the option to download the information in PDF and CSV format.
   ii. The system must have the option to export the schedule to Outlook and Google calendars.
   iii. The system must be have the option to be integrated (embedded) to the current University portal, if required (SSO).
   iv. The system must have the option to be integrated to standard directory protocol (LDAP)

c. Normative
   i. The system's information must be archived for a period of 4 years.
   ii. The system must complain the current Quebec normative for educational web applications.

2. Reliability

a. Availability
   i. Since it is not mission critical, the system must have an annual uptime of 99.9% (i.e. the maximum annual downtime must be 87.6 hours)
b. Recoverability
   i. On case of system failure, the maximum restore time must be 4 hours.
   ii. On case of database failure, the maximum information lost must be 1 day.

3. Usability

a. Understandability
   i. The system's web user interface should be designed so the users do not require any other assistance (manual, external applications) to operate it (user friendly)
   ii. The system must be offered in English and French. I must have to option to change its language.

b. Accessibility

i. To grant access to senior people, the minimum system's font size will be 14px, along with soft, no high contrast colors.
ii. To grant access to blind people, the system should have menu voice assistance and screen readers (visually impaired user's technology).

4. **Efficiency**
   a. Time
      i. The system's response time will be less than 3 seconds in all operations.
   b. Resources
      i. The system must designed to run on medium size server (4-8 Cores, 16-32 GB RAM)
5. **Maintainability**
   a. Scalability
      i. The system must be able to support 5.000 users, and allow a grow up to 20.000 users.
      ii. The system must be able to serve 500 concurrent users.
   b. Support
      i. To help support, technical documentation about the system use, design, installation, configuration and troubleshooting must be created, and the source code must be documented.
   c. Testability
      i. Evidence about the test cases performed during its development must be provided.
6. **Portability**
   a. Adaptability
      i. The system must adapt its interface for both laptop/desktop browser and mobile devices with screens 8 inch or bigger.
      ii. The system must work on the latest versions of Chrome and Firefox.
      iii. The system must be build using modern web standards (HTML5, CSS3).

4. RESOURCE EVALUATION

In this section, the resources available in the building of the Apollo application will be provided. Human resources, in terms of each member's capabilities and their expected contributions in the building of this software will be looked at, as well as the technical resources available to the team.

4.1 Human Resources

Team Member: **Philippe Abou Kasm**
Primary Role: Documentation/Testing
Capabilities: I took COEN 243/244, which I have knowledge in object oriented programming in C++. I also took COEN 352, which I have knowledge in data structures and algorithms and familiarity with Java programming. Finally, I took COEN 311, with basic knowledge in assembly language programming. I have basic knowledge in HTML.
Availability: I am willing to contribute 5 hours per week meeting included. I am also available sometimes on weekends.

Team member: **Wahab Ahmed**
Primary Role: Documentation
Capabilities: I took SOEN 287 a year ago, which we learnt HTML, PHP, JS, and briefly touched databases for the final website assignment. I worked as a mechanical CAD drafter at some Toronto based company.
Availability: I can contribute 5 hours to the project weekly, meetings included. I am available to meet any time that doesn't clash with my class schedule, including weekends.

Team member: **Sabrina Ashraff**
Primary Role: Documentation/Testing
Capabilities: I did SOEN 287, which I learned HTML5, CSS, Javascript and PHP. I did two co-op internships as an analyst programmer at CMC Electronics, customizing their customer resource management application to fit their needs as well as writing up documentation on where the customizations were made and creating instruction guides for non-technical users.
Availability: I can contribute around 4 hours per week, including meeting. I may miss class occasionally as I am currently searching for another internship under the co-op program. Weekend availability may vary as I work on weekends.

Team Member : **Francis Bouchard**
Primary Role: Programmer
Capabilities: I took SOEN 287 (Web programming class) where we learned various languages such as HTML, CSS, Javascript and PHP. I had a 4 months of co-op internship as a web developer at UQAM. I also developed RESTful APIs to replace legacy applications. I have experience in Node.js and Angular. I have good understanding of various web development design patterns.
Availability: I can commit to 4 hours per week on this project including meetings. I may miss classes and meetings occasionally because of job interviews as I am currently looking for a second co-op internship for the summer.

Team member: **Clozzy-Mailet Chavez**
Primary Role: Documentation
Capabilities: I took COEN 243/244/352, which I learned object oriented programming in C++, and a bit of Java, respectively. I also too ENGR 290/301, which I learned about project management, technical writing and technical presentation in a team environment. I have basic knowledge in HTML.
Availability: I can commit to about 5 hours per week, meetings included.

Team Member: **Ricardo Cortes**
Primary Role: Documentation
Capabilities: I am a Electronic/Computer Engineering undergraduate and I am currently doing my M. Eng. Software Engineering. I took SOEN 6471, which teaches advanced software architectures, took SOEN 6481, a class on system requirement specifications, and SOEN 6461, which is software design methodologies. I have several years of work experience, in which I had performed different roles. I have had the opportunity to work in several project, at different phases, both development (new) and maintenance (existing), in different industries, with different methodologies. I can contribute to the project with my experience. I am good with Java, Design Patterns, Architectural Patterns.
Availability: I am available Monday to Friday only, about 4 hours per week, including meetings.

Team Member: **Liuai Hatter**
Primary Role: Programmer/Leader
Capabilities: I took SOEN 287, but have basic knowledge of PHP. I took ENGR 301 (Eng. Management Principles and Economics), which I learned MS Project, work breakdown schedule, and Gantt charts. I am a graduate from Bachelor of Fine Arts and have worked as a graphic designer (all for web) for a tech company.
Availability: I would say about 5 hours a week (meetings included). I am also available to meet during weekdays, evenings, and sometimes weekends.

Team Member: **Jian Huang**
Primary Role: Programmer
Capabilities: I took SOEN 287, which I learnt HTML5, CSS, Javascript and a bit of PHP. I also took ENGR 301, which is a management course for engineering, where I learnt a lot of processes.
Availability: I can contribute to as many house (5 or more hours) as possible to the project as I only have assignments to do, as long as it does not interfere with my class schedule.

Team Member: **Anna Rogozin**
Primary Role: Programmer
Capabilities: I completed the Computer Science Technology program at Dawson College, where I had a few web programming classes similar to SOEN 287. I have gained experience through internships in the past and currently am on an internship dealing with machine-to-machine communication. I'm familiar with different design patterns and approaches to software development, which can be useful for this project.
Availability: I can contribute up to 5 hours per week, meeting included, but I prefer in the evening or during the weekend.

Team Member: **Ramy Sandouk**
Primary Role: Documentation
Capabilities: I am a second year student in computer engineering. I took COEN 243, COEN 244, COEN 352 and COEN 311, which I learned C++, Data Structures and Computer Organization, respectively.
Availability: I can contribute 4 to 5 hours per week.

Team Member: **Matthew Teolis**
Primary Role: **Programmer**
Capabilities: I obtained my DEC in the Computer Science Technology program at Vanier College. I have a fairly good understanding of how the RESTful API design pattern works. I've worked with PHP, Git, MVC pattern, and the Laravel framework for one year in the workforce, including my internship time. I am a good and efficient coder
Availability: I am available 5 hours per week, including meeting, minimum.

4.2 Technical Resources

The following technical resources will be used in the development of the Apollo application:

- Multiplatform (windows, mac, linux)
- Frontend: angular material framework (to implement the material design)
- Backend: PHP with laravel framework, database with MySQL, data structures are with JSON.
- IDE: PHP Storm, Notepad++
- Gulp
- WAMP / MAMP / XAMPP

In terms of collaboration and communication between team members, the following technologies will be used:

- Github
- Slack
- Google Drive

5. SCOPING

After careful evaluation of the resources available to develop the application, some features that had been initially been anticipated to be included have been left out. The following requirements have been scoped out of the project:

1. System Administrator

1.1. System administrator can create new users
  1.1.1. System administrator can grant new users specific permissions
1.2. System administrator can view existing users
1.3. System administrator can update existing users' information
  1.3.1. System administrator can change users' permissions
1.4. System administrator can remove existing users from the system

Reason: Unable to implement due to time constraints

2. Faculty Administrator

2.1. Faculty administrator can create new courses
  2.1.1. Faculty administrator can assign requisites and requirements to courses
2.2. Faculty administrator can view existing courses
2.3. Faculty administrator can update existing courses' information
  2.3.1. Faculty administrator can change course requisites and requirements
2.4. Faculty administrator can remove existing courses from the system

Reason: Unable to implement due to time constraints

3. Professor

3.1. Professor can view his or her course schedule
3.2. Professor can update students' grades

Reason: Unnecessary for general functionality of the system

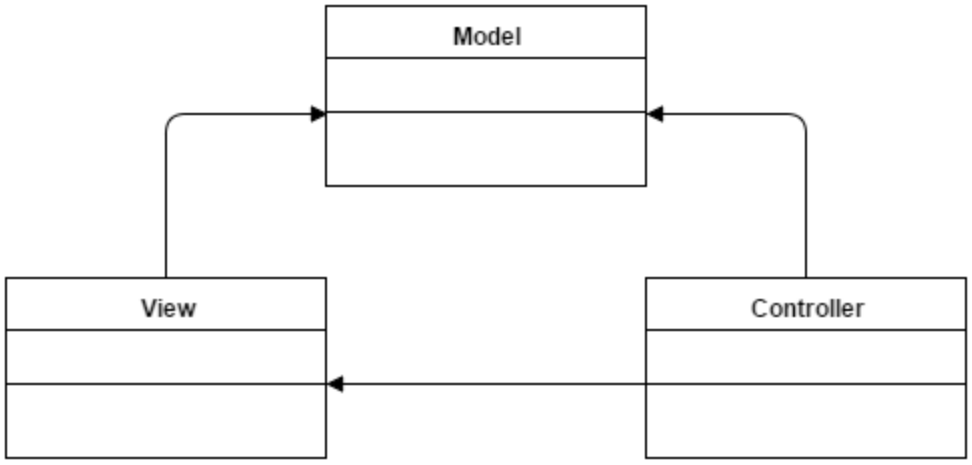
4. Student

4.1. Student can generate schedules
  4.1.4. Student can finalize a schedule

Reason: Unnecessary for general functionality of the system

6. SOLUTION SKETCH

6.1 Architecture



The Apollo project will require the use of the Model-View-Controller (MVC) architecture design pattern. This pattern suits best for the development of our system because it contains distant separation between domain, the actions based on user inputs, and the presentation. This will ease up the development phase as programmers will be able to work on three different classes at the same time. The model will be the one that manages the behaviour of data, responds to information about the state, and responds to instructions to change state. To enforce this pattern, we will be using Laravel, a framework for developing web applications. The view, on the other hand, is a visual representation of the model. It will get information from the model and display it appropriately. The view will therefore need to know the semantics of the attributes of the model. We will be using Angular.js, a JavaScript framework, as well as Angular Material framework to implement a beautiful design for the user to interact with. The controller handles user interaction. The Controller will thus read data from the view, will then inform and make change to the model if desired, by controlling user input. As seen in the diagram, both the view and the controller depend on the model. However, the model does not depend on neither of them. This is one great advantage of this design architectural pattern because it allows the model to be developed and tested without having to make changes to the view.

All the communication between the backend and the front end will be done in a RESTful web service format. This means that the application communicates by

exchanging resources back and forth using specific HTTP requests. For example, when someone wishes to store a course in the database, the user will create a JSON with all the course's data inside. Then, the POST http method will be used on the url http://apollo.matthewteolis.com/api/v1/courses, with the json inside the payload. This will be interpreted by Laravel and stored in the database.

6.2 Technologies in Use

PHP (Hypertext PreProcessor)

PHP is a web scripting language. The application is a web application which explains why PHP has been chosen as the language. The server will be running off of PHP scripts, in conjunction with Laravel (which is a PHP framework). PHP is an easy to learn and a very powerful language. It will make the project development quick and easy.

Laravel

Laravel is an open source PHP framework. It follows the MVC pattern using their own blade compiler. This allows developers to quickly deploy web projects. Laravel's Eloquent, which is uses the Object-relational mapping (ORM) technique, makes database manipulations really easy because it can access the database as if they were their own objects. Creating and populating databases to different machines is also easily done with the database migration commands they offer, they made their own database version control. Laravel has the RESTful design pattern integrated with the resource controller, therefore you can create a new REST URL in 2 lines.

JetBrains PHPStorm

The reason we chose to go with PHPStorm is because it has a lot of integrations with the IDE. The IDE has integrations such as Git, Vagrant, Composer, debug with a live server, has good code sniffing and intellisense, use the terminal (right in the IDE), Gulp, NodeJS, Laravel support, make SSH connections, even use RESTful requests. PHPStorm is very powerful tool that makes the development process faster and easier.

Gulp

Gulp is a minifying application. Gulp will be obfuscating and minifying all css and javascript files in the project. There is also a watch feature that automatically minifies and combines the javascript and css files when the user saves the file, which makes it quick for development.

Angular.js

AngularJS is an open source javascript framework. It binds data to HTML with expressions and extends HTML attributes with directives. Angular's code is downloaded and included in the html files.

Angular will be used to perform the REST API calls on the front-end of our applications. It will communicate with the Laravel back-end. Angular Material framework will be used to implement the material design, this will give our website a easy to use, beautiful user interface.

HTML

HTML (HyperText Markup Language) is a markup language used to display content from the web. Any web app must use HTML to be able to interact with the user. HTML uses tags to describe every element on the web page.

JSON

JSON stands for Javascript Object Notation. It is a popular format used to store and exchange data. It is an alternative to the XML format. We will be using this format because it is easier to use and more readable than XML. All the information we collect and send to the user will be stored in a JSON.

Git

Git is a code repository and Github is a git hosting service. All code and documentation is maintained using Github. Using it for documentation with Google Drive is redundant; but not every team member is familiar with Github, therefore at this point in the project documentation is maintained on both systems. The benefits of Github as a repository for documentation and code is that issues can be made, team members assigned to the issue, and the issue can be tracked until resolved. Then the code or documentation can be pushed onto the repository by the issue handler.

Google Drive

Google Drive is a repository for documents. It does not offer issue tracking but has the advantage of being easy to learn. All documents can be edited by a member of the team.

Slack

Slack is a cloud based collaboration software. It enables the creation of many channels and allows for group and individual communications. Channels can also be

linked to other tools and software. For example, a #calendar channel was created where group meetings are organized and scheduled; it is also directly linked to the group's Google Calendar. The #calendar channel on Slack is updated and group members alerted anytime the Google Calendar is changed. There are also a #general channel where aspects of the project are discussed, a #useful channel where important links and reminders are posted. Channels can also be made for subgroups, such #softwareteam for the programming sub-group. Alerts can specified by each team member: to receive an email notification at every update in Slack, or only for personal messages, or notification after only a certain period of time. There is also a Slack desktop app which can notify users directly.

MySQL

MySQL is an open-source relational database management system (RDBMS). It is widely used for both small and large web-based applications. We will be using MySQL due to its high-performance and reliability. Its cross-platform support suits the many operating systems that we will use. In addition, MySQL combined with PHP (our chosen programming language) allows rapid web application development.

Apache Server

The Apache HTTP Server is a powerful open-source tool that provides HTTP services and integrates well with PHP. It is a secure, efficient, and extensible web server that is easy to manage and is available for all the platforms used in our project.

XAMPP

XAMPP is a web server solution pack that stands for cross-platform Apache HTTP Server MySQL PHP Perl. It is a portable development environment that introduces consistency to a team working on multiple platforms and that allows testing on local machines. XAMPP includes the previously mentioned PHP, MySQL, and Apache Server technologies, thus enabling us to decrease development setup time.

7. Plan

7.1 Activities

Research	
Description	Research for technologies in use (eg: frameworks, design patterns, design specifications etc). Requirements research (eg: classrooms, requisites, professors).
Artifacts	15, 26

Discussion	
Description	Multiple weekly meetings (in person and online). Assigning various tasks to team members and discussing solutions and approaches to our application.
Artifacts	1, 2, 4, 5, 12, 13, 28

Planning & Designing	
Description	Planning out how to implement the application design (eg: drawing database architecture, domain model, uml diagrams, frontend design, dynamic design scenarios etc.)
Artifacts	1, 3, 4, 5, 20, 21

Technical Configuration	
Description	Setting up our framework, version control, slack communication, google drive. Setting up the frontend and backend communication. Setting up the database.
Artifacts	6, 7

Data Input	
Description	Extracting course information and inputting course data into the database.
Artifacts	9

Implementation	
Description	Importing the framework modules. Creating RESTful API. Implementing all features.
Artifacts	10, 14, 18, 19

Testing & Debugging	
Description	Creating a test plan. Testing application for functionality and checking if features are working correctly. Verifying bugs are fixed.
Artifacts	23, 24

Documentation	
Description	Documenting all planning, specifications, design, and issues encountered. Creating the user manual. Documenting source code.
Artifacts	3, 8, 11, 16, 17, 20, 22, 25, 27, 28

7.2 Artifacts

1	Member Organization
Team members subdivided themselves into groups of programmers and documenters according to skill set.	

2	Initial Discussion
The initial discussion involves the gathering of ideas and information around the project. Initial software requirements and designs were proposed. Team members shared their key strengths in order to participate effectively in the process. A brainstorming of the structural and functional requirements of the scheduler lead to the establishment of a foundation for the software development process.	

3	Deliverable 0
Deliverable 0 consisted of all the elements that were discussed in the initial discussion. A description for the project was provided, along with the domain model diagram and a description of the areas within the domain model. An initial list of the team members and the tasks they were interested in accomplishing regarding the application was also drawn up.	

4	Scheduling of Tasks
This artifact involved assigning specific tasks to each member of the team regarding the project. Meeting times for each subgroup working on a certain task were also arranged and set up, as well as dates as to when each task should be expected to be completed.	

5	Resource Evaluation
Establishing everyone's technical capacities and the amount of time each individual can spend working on the project. Collected every technical resource available to the team including software tools and computers.	

6	Framework Configuration
Downloading and installing laravel on each programmer's machine. Importing the various libraries needed to make run the application through the use of composer. Importing the database.	

7	Collaboration Tools
Setting up every team member with all collaboration tools. Creating a slack team to set up meetings, to discuss and to communicate useful information. Creating a Google Drive for all team members to contribute documents, in order to work on collaboratively. Creating a GitHub organization for all teams members to access the repository which includes documents and codes. This allows to keep track of any revisions and issues that occur.	

8	Domain Model Design
Creating the domain model diagram to document the key concepts and terminology of the system. It consists of domain level objects, their attributes and associations.	

9	Course Database
Adding to the course database the course titles, professors, time slots, requisites, descriptions, credits and faculty of each course required for the software engineering degree. Inputting the course lecture, tutorial, and lab time slots.	

10	Prototype
Creating a live prototype to allow team members to input course information to the database. The prototype will demonstrate the integration of laravel and angular.js frameworks for our application.	

11	Use Case Tables & Diagrams
The use case tables describe in detail every possible user (actor) interaction with the system. The use case diagrams visualize these interactions, grouped by actors. This artifact is important for the identification of functionalities of the system.	

12	Constraints Identification

This artifact involves analyzing and classifying any constraints that the team might have. It includes design, hardware, performance, security, or external constraints. The identified constraints are then described in detail.

13	Scoping

The scoping artifact describes the functionalities that are removed from the final system design due to limited time or resources. The functionalities may include features, goals, or requirements and qualities of the system. For each scoped out functionality, a clear reason is given.

14	Initial Front End Planning & Design

Establishing design patterns to follow when creating the user interface. Choosing a color palette. Drawing mock-ups.

15	Estimation & Risk Planning

Carefully identifying risks that the project can have so team members can take pre-caution to avoid those risks. Estimating the total cost of the project after carefully estimating the cost of production of each artifact. Also estimating the time for each task and scheduling accordingly.

16	Deliverable 1 Documentation

Deliverable 1 consisted of the initial project description and domain model which was provided in deliverable 0. In addition to that, after several team meetings we defined the goals of our project in the document along with all the constraints, some of which were scoped down later on. Human and technical resources were discussed and tasks were split accordingly. An early overview of our solution and plan along with a prototype was provided in this document.

17	Architecture Design & Diagrams
An updated and detailed version of the architectural design. It includes a description of the reasons behind our design, the changes we made from the previous design in Deliverable 1, and all the components (such as function calls and description of parameters) included in our new design. This will be presented in the form of UML diagrams for our complete description of the system design.	

18	Frontend Implementation
Development of the frontend of our system, the part where the user can interact with.	

19	Backend Implementation
Development of the backend of our system, which includes the server, the application and the database.	

20	UML Class Diagrams
Create Unified Modelling Language (UML) class diagrams for all classes of program. Responsible person who meet with developing team.	

21	Dynamic Design Scenarios
A full dynamic design of use cases including system sequence designs, operational contracts and sequence diagrams.	

22	Deliverable 2 Documentation
Deliverable 2 consists of a much more structured and detailed design for our system, including better UML class diagrams. It also includes an updated, and more a detailed version of the architectural design from the previous one. Dynamic design diagrams are also well put together. This deliverable will help the programmers give a more organised overview of the system, which they can report on its rapid prototypes and examine the effects on estimates, risks and scopes.	

23	Test Planning & Testing
Meet with the programmers and discuss the program's possible weaknesses, compile a list. Meet with the testing team and plan out use cases specific to weakness discussed with developers; also plan other cases covering all other aspects the testing team can derive. Compile use cases, test them on program and record success, failures, and comments.	

24	Improvements
Implement risk management for all problems found during testing; treat problems found as identified risks. Implement risk analysis (prioritizing), risk planning (mitigation, avoidance, and reprogramming), and create document for risk monitoring to be part of system administrator's user manual.	

25	User Manual
Create User manual for end user and system administrator.	

26	Final Cost Estimate
A final listing of all components of all phases of the project, including the hours cost per person for each of the components of each phase.	

27	Deliverable 3 Documentation
Deliverable 3 consists of documenting the testing process, including unit, requirement, stress and security testing. Includes the user manual, and final cost estimate.	

28	Project Documentation & Presentation
This includes the final and complete documentation of the project. A presentation is given to demonstrate the fully functional application.	

7.3 Project Estimation

Tasks	Project Estimate (Hours)
Member Organisation	1
Initial Planning	2
Deliverable 0	4
Scheduling	2
Resource Evaluation	5
Technical Configuration	3
Domain Model Design	3
Architecture Planning	5
Initial Architecture Design	3
Database Setup	24
Rapid Prototype	24
Data Input	15
Functional Requirements Planning	10
Constraint planning	5
Scoping	5
Initial Front End Planning	5
Estimation 1	1
Risk Planning 1	1
Deliverable 1	30
Architecture Design	5

Front End Planning	10
Front End Design	10
Detailed Design	10
Dynamic Design Scenarios	20
Front End Prototype	30
Estimation 2	1
Risk Planning 2	3
Deliverable 2	30
Implementation Planning	15
Implementation	40
Test Planning	15
Testing	30
Improvement	15
Testing	10
User Manual	20
Final Cost Estimate	2
Deliverable 3	40
Project Documentation	80
Deliverable 4	15
TOTAL	**549**

Project Re-estimation Scenarios

Front End Prototype

Re-estimation of the hours costed may occur due to unforeseen circumstances that may occur in the creation of the prototype. Some parts of may take more time than anticipated if it does not function the way it is intended, or if we run into bugs in initial stages. Similarly, it may require less time than anticipated if everything runs smoothly.

Implementation

Re-estimation of hours costed may occur while implementing features and writing code. This may include initial bugs that may immediately occur and must be resolved, as well as running into problems implementing certain functionalities that may not have been predicted. Similarly, it may cost less time if everything runs smoothly without issue.

Testing

Re-estimation may be necessary as the duration of the testing phase depends on how well the initial application functions. If there are more problems and bugs than anticipated, the testing phase may take longer as more tests will be required to point out any malfunctioning areas.

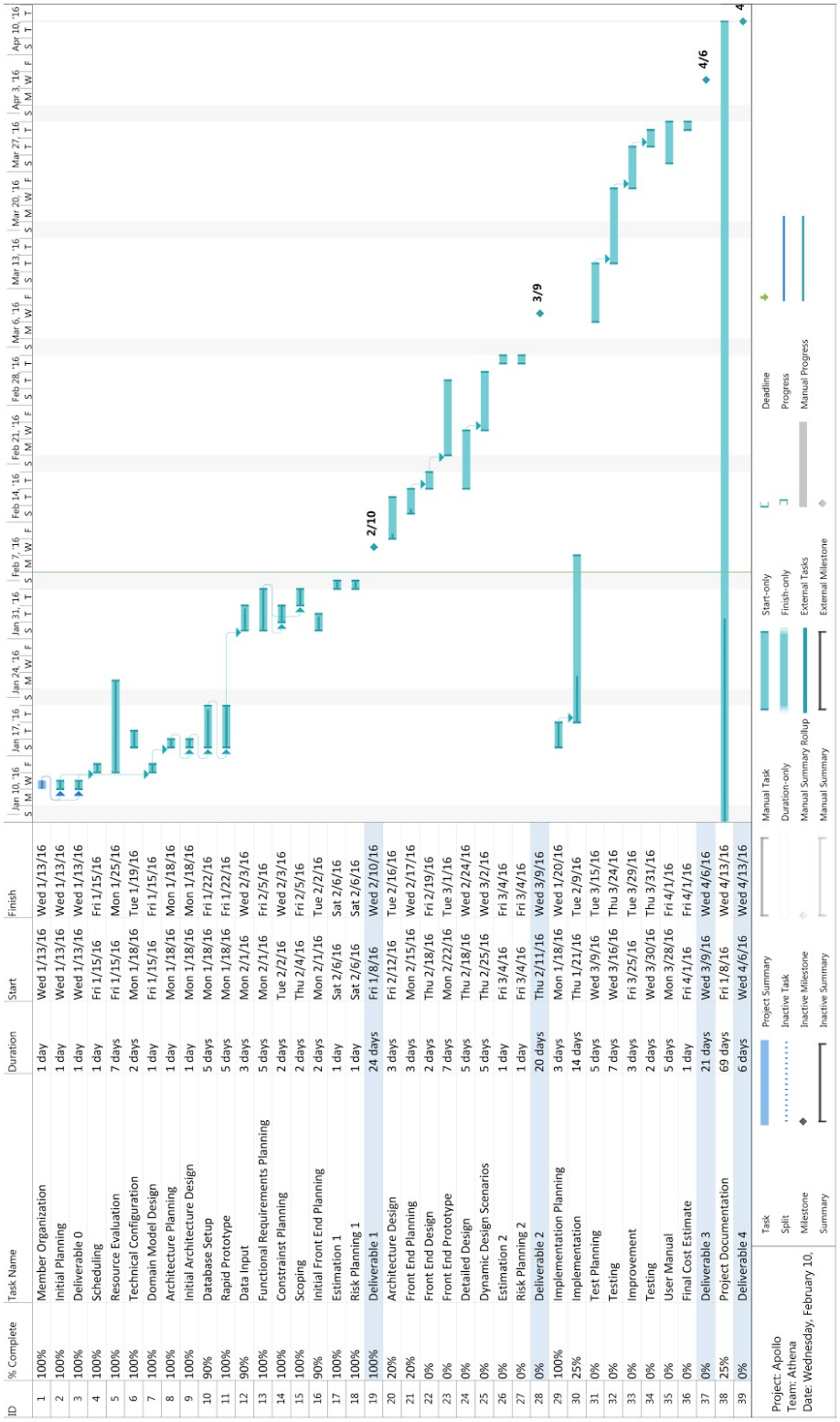
Improvement

Re-estimation may be necessary as the improvement phase is largely dependent on how well the initial application functions. It may require longer if more problems and bugs than anticipated are found and need to be fixed.

7.4 Activities Assignment

Team Member	Activities Assignment
Philippe Abou Kasm	Data input, Plan (7.1, 7.2, 7.4, 7.5), Documentation, Testing
Wahab Ahmed	Plan (7.3, 7.6), Documentation, Testing
Sabrina Ashraff	Presentation, Project description, Plan (7.1, 7.2, 7.4, 7.5), Programming
Francis Bouchard	Solution Sketch (6.2), Plan (7.1, 7.2, 7.4, 7.5), Prototyping, Programming (Front-end)
Clozzy Chavez	Data input, Plan (7.1, 7.2, 7.4, 7.5), Presentation, Documentation, Testing
Ricardo Contés	Goals & Constraints, Scoping, Documentation
Liuai Hatter	Lead, Goals & Constraints, Scoping, Plan (7.1, 7.2, 7.4, 7.5), Presentation, Programming (Front-end)
Jian Huang	Solution Sketch (6.1), Plan (7.3, 7.6), Programming
Anna Rogozin	Goals & Constraints, Scoping, Solution Sketch (6.2), Programming (back-end)
Ramy Sandouk	Solution Sketch (6.2), Documentation, Testing
Matthew Teolis	Solution Sketch (6.2), Prototype, Data input, Programming (front-end & back-end)

7.5 Schedule



ID	% Complete	Task Name	Duration	Start	Finish
1	100%	Member Organization	1 day	Wed 1/13/16	Wed 1/13/16
2	100%	Initial Planning	1 day	Wed 1/13/16	Wed 1/13/16
3	100%	Deliverable 0	1 day	Wed 1/13/16	Wed 1/13/16
4	100%	Scheduling	1 day	Fri 1/15/16	Fri 1/15/16
5	100%	Resource Evaluation	7 days	Fri 1/15/16	Mon 1/25/16
6	100%	Technical Configuration	2 days	Mon 1/18/16	Tue 1/19/16
7	100%	Domain Model Design	1 day	Fri 1/15/16	Fri 1/15/16
8	100%	Architecture Planning	1 day	Mon 1/18/16	Mon 1/18/16
9	100%	Initial Architecture Design	1 day	Mon 1/18/16	Mon 1/18/16
10	90%	Database Setup	5 days	Mon 1/18/16	Fri 1/22/16
11	100%	Rapid Prototype	5 days	Mon 1/18/16	Fri 1/22/16
12	90%	Data Input	3 days	Mon 2/1/16	Wed 2/3/16
13	100%	Functional Requirements Planning	5 days	Mon 2/1/16	Fri 2/5/16
14	100%	Constrainst Planning	2 days	Tue 2/2/16	Wed 2/3/16
15	100%	Scoping	2 days	Thu 2/4/16	Fri 2/5/16
16	90%	Initial Front End Planning	2 days	Mon 2/1/16	Tue 2/2/16
17	100%	Estimation 1	1 day	Sat 2/6/16	Sat 2/6/16
18	100%	Risk Planning 1	1 day	Sat 2/6/16	Sat 2/6/16
19	100%	Deliverable 1	24 days	Fri 1/8/16	Wed 2/10/16
20	20%	Architecture Design	3 days	Fri 2/12/16	Tue 2/16/16
21	20%	Front End Planning	3 days	Mon 2/15/16	Wed 2/17/16
22	0%	Front End Design	2 days	Thu 2/18/16	Fri 2/19/16
23	0%	Front End Prototype	7 days	Mon 2/22/16	Tue 3/1/16
24	0%	Detailed Design	5 days	Thu 2/18/16	Wed 2/24/16
25	0%	Dynamic Design Scenarios	5 days	Thu 2/25/16	Wed 3/2/16
26	0%	Estimation 2	1 day	Fri 3/4/16	Fri 3/4/16
27	0%	Risk Planning 2	1 day	Fri 3/4/16	Fri 3/4/16
28	0%	Deliverable 2	20 days	Thu 2/11/16	Wed 3/9/16
29	100%	Implementation Planning	3 days	Mon 1/18/16	Wed 1/20/16
30	25%	Implementation	14 days	Thu 1/21/16	Tue 2/9/16
31	0%	Test Planning	5 days	Wed 3/9/16	Tue 3/15/16
32	0%	Testing	7 days	Wed 3/16/16	Thu 3/24/16
33	0%	Improvement	3 days	Fri 3/25/16	Tue 3/29/16
34	0%	Testing	2 days	Wed 3/30/16	Thu 3/31/16
35	0%	User Manual	5 days	Mon 3/28/16	Fri 4/1/16
36	0%	Final Cost Estimate	1 day	Fri 4/1/16	Fri 4/1/16
37	0%	Deliverable 3	21 days	Wed 3/9/16	Wed 4/6/16
38	25%	Project Documentation	69 days	Fri 1/8/16	Wed 4/13/16
39	0%	Deliverable 4	6 days	Wed 4/6/16	Wed 4/13/16

7.6 Risks

Use of an unfamiliar framework:

- Not everyone in the team is knowledgeable about the Laravel framework being used for the project.

- Amateur developers might face problems while extending code and classes, as it is a new platform for most of the developers to deal with.

- Because the team is working with a new framework, the scheduling part of the project is a hard task to do and learning how to work with the framework could delay the assigned tasks.

Different levels of programming knowledge:

- Not everyone in the team comes from a software engineering background. Different levels of experience with the programming language could result in inability to understand complex pieces of code for some team members.

- Not everyone necessarily follows commenting guidelines. One developer might have coded something without proper documentation about it which could be confusing for other developers working on the same project resulting in delays and other problems.

Schedule generating issues:

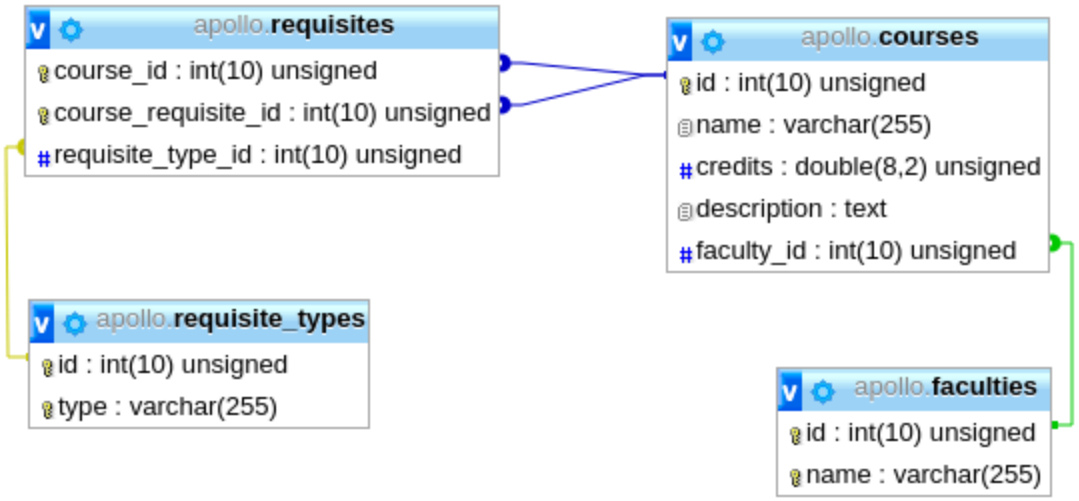
- Students might end up with a course in their schedule that they shouldn't be allowed to take according to university's rule and regulations. For example, a student is registered for a regular Fall/Winter year, and he fails a course in fall which is a prerequisite for some other course in winter (or gets a D+ or lower in some 200 level course). With this issue the student could still be enrolled in the course in winter even after failing its prerequisite which could violate the university's regulations.The scheduler might show a specific class full before its full capacity has been reached, possibly resulting in graduation delay for many students. For example, if a class like SOEN 342 or 343 is shown full when it is actually is not, a student will have to wait one full academic year to be able to take it as it's offered only once every year.

Security issues:

- Security is a huge problem when it comes to storing confidential data about the users in a database. Security measures such as having a secured server to handle the information and much testing is needed before the final release of our software.

8. Prototype

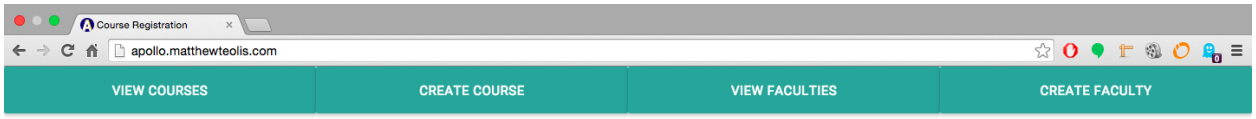
We are using a RESTful API using laravel to connect the backend to the frontend. We can create, delete, list all courses and show specific courses and faculties. We display this information using Angular.js, and the Angular Material framework. Angular send AJAX requests to the backend and receives a JSON containing the information requested. We have a working database for the courses, prerequisites and faculties. The following diagram shows the structure of the current database for the prototype:



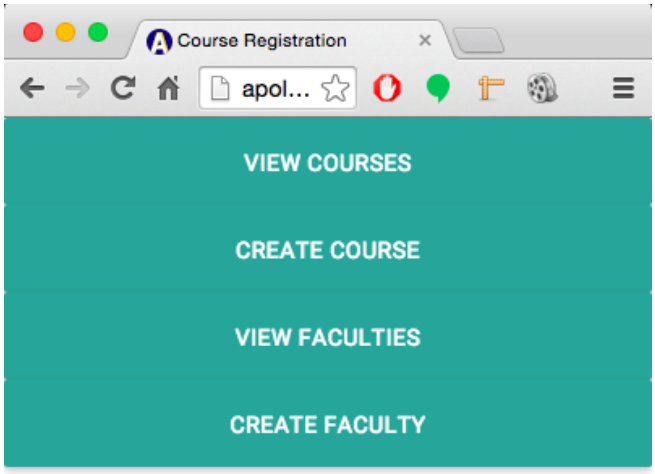
The prototype website has been used in order to populate all the courses. The data has been taken from Concordia's course list.

URL: http://apollo.matthewteolis.com/
Username: soen
Password: 341

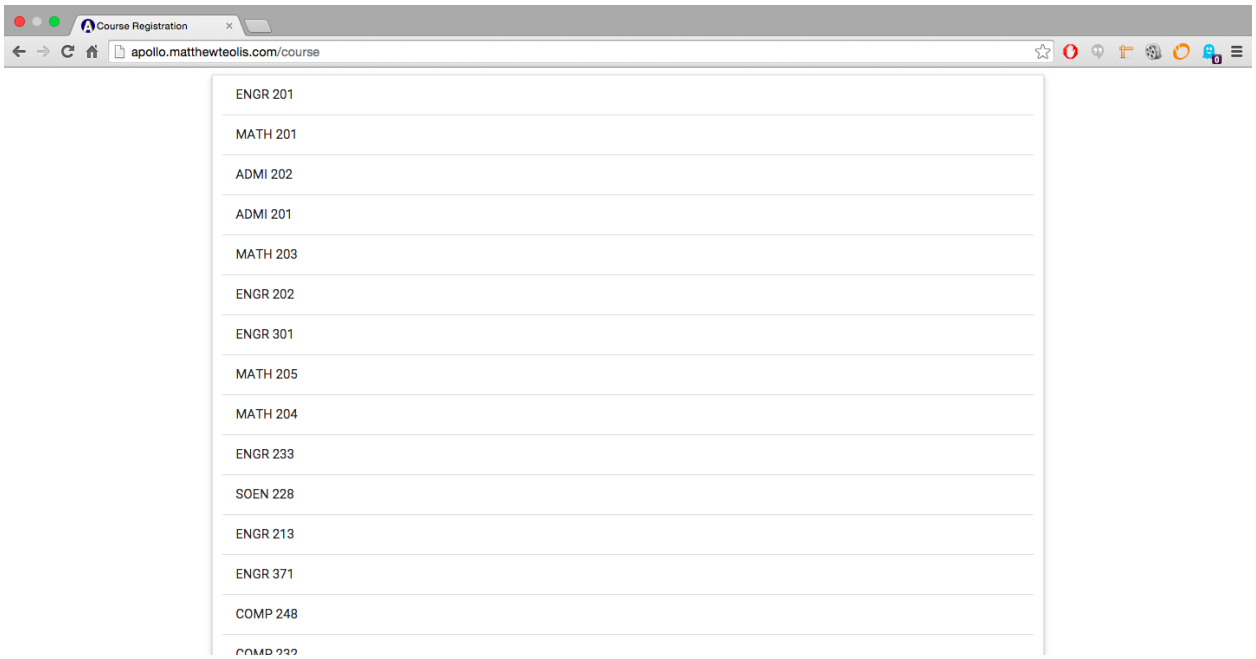
Full Width Screenshot:



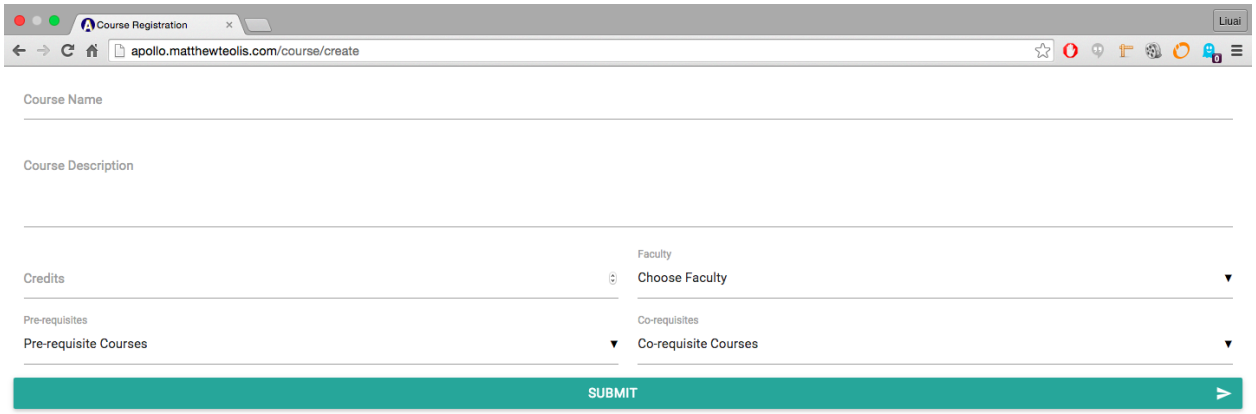
Mobile Width Screenshot:



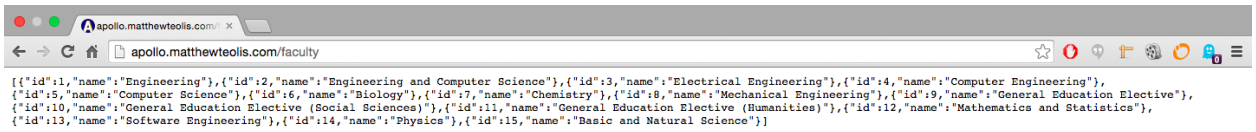
View Courses Page:



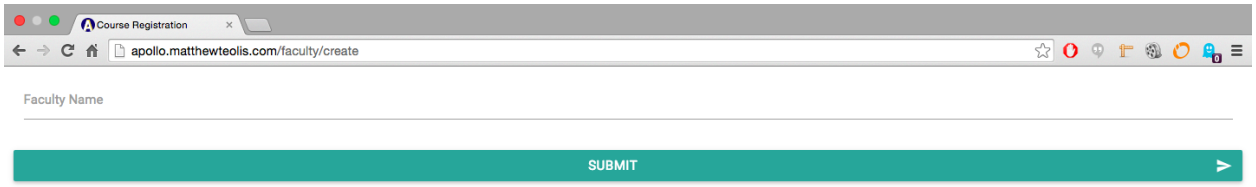
Create Course Page:



View Faculties Page:



Create Faculty Page:



RESTful Request Table:

Verb	Path	Action
POST	api/v1/course	Inserts a new course in the database.
GET	api/v1/course	Retrieves all courses from the database.
GET	api/v1/course/{id}	Retrieves a specific course with a matching id parameter.
PUT	api/v1/course/{id}	Updates a specific course with new attributes.
DELETE	api/v1/course/{id}	Deletes a specific course with a matching id parameter.

NOTE: Please refrain from altering the data.

Storing a new course to the database:

```php
/**
 * Store a newly created resource in storage.
 *
 * @param  \Illuminate\Http\Request  $request
 * @return \Illuminate\Http\Response
 */
public function store(Request $request)
{
    $input = $request->input();

    $course = new Course;
    $course->name = $input['course_name'];
    $course->credits = $input['course_credits'];
    $course->description = $input['course_description'];
    $course->faculty_id = $input['faculty_id'];
    $course->save();

    // Pre-requisite is id 1 in RequisiteType table
    if(isset($input['prerequisites']))
        Requisite::addRequisites($course->id, $input['prerequisites'], 1);

    // Co-requisite is id 2 in RequisiteType table
    if(isset($input['corequisites']))
        Requisite::addRequisites($course->id, $input['corequisites'], 2);

    return response()->json($course);
}
```

Using the RESTful API conventions, listed in the previous RESTful request table , the store method listens for a “POST” request at the “api/v1/course” route.

Course Model:

```php
class Course extends Model
{
    /**
    * Whether or not the table uses timestamps
    */
    public $timestamps = false;

    /**
    * This will hide the following fields when getting a query result
    */
    protected $hidden = [
        'faculty_id',
        'pivot'
    ];

    public function faculty()
    {
        return $this->hasOne('apollo\\Models\\Faculty', 'id', 'faculty_id');
    }

    public function prerequisites()
    {
        return $this->requisites(1);
    }

    public function corequisites()
    {
        return $this->requisites(2);
    }

    private function requisites($type_id)
    {
        $instance = new Course;
        $relation = $this->getBelongsToManyCaller();
        $foreignKey = 'course_id';
        $otherKey = 'course_requisite_id';
        $table = 'requisites';
        $query = $instance->newQuery()->where('requisite_type_id', $type_id);
        return new BelongsToMany($query, $this, $table, $foreignKey, $otherKey, $relation);
    }
}
```

The course model access the database directly. By default it will access the “courses” table because the name of the model is “Course” (it adds an “s” at the end of the model name). Therefore, creating new courses and saving them to the database can be done easily.

9. Works Cited

[1] ISO/IEC 9126-1:2001 Software engineering - Product Quality, http://www.iso.org/iso/catalogue_detail.htm?csnumber=22749

[2] "Domain Model", class notes for SOEN 341 Software Process, Department of Computer Science and Software Engineering, Concordia University, Quebec, Winter 2016.

[3] "The Software Process", class notes for SOEN 341 Software Process, Department of Computer Science and Software Engineering, Concordia University, Quebec, Winter 2016.

[4] "Requirements and User Case", class notes for SOEN 341 Software Process, Department of Computer Science and Software Engineering, Concordia University, Quebec, Winter 2016.

[5] "Risk Management", class notes for SOEN 341 Software Process, Department of Computer Science and Software Engineering, Concordia University, Quebec, Winter 2016.

[6] "Activity Planning", class notes for SOEN 341 Software Process, Department of Computer Science and Software Engineering, Concordia University, Quebec, Winter 2016.

[7] "Estimation Techniques", class notes for SOEN 341 Software Process, Department of Computer Science and Software Engineering, Concordia University, Quebec, Winter 2016.

[8] "Architecture and Design", class notes for SOEN 341 Software Process, Department of Computer Science and Software Engineering, Concordia University, Quebec, Winter 2016.

[9] T. Otwell. *Laravel* [Framework]. Available: https://laravel.com/

[10] Brat Tech LLC, Google. *AngularJS* [Framework]. Available: https://angularjs.org/