**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**

**1. 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.

**2. 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.

**2.1 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

1. **System Administrator**

    2.1. System administrator can create new users

        2.1.1. System administrator can grant new users specific permissions

    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.

1. **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

1. **Professor**

    4.1. Professor can view his or her course schedule

    4.2. Professor can update students’ grades

1. **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

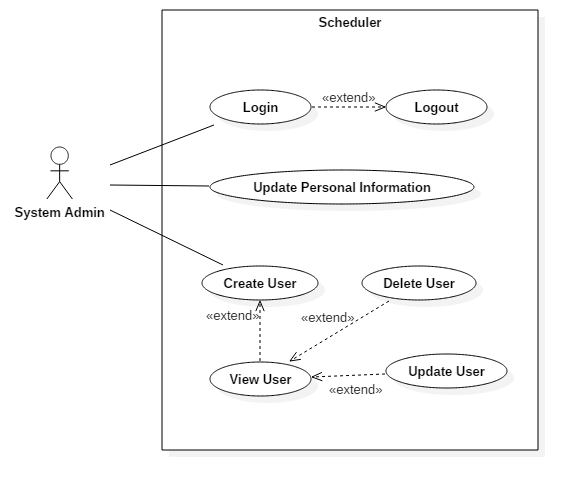
        5.4.3. Student can view saved schedules

        5.4.4. Student can finalize a schedule

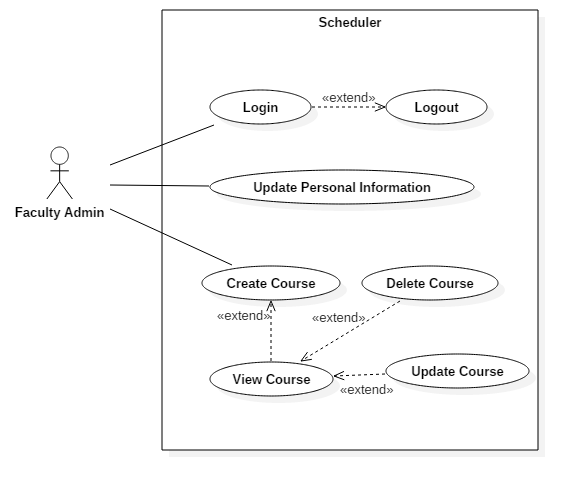
    5.5. Student can print a schedule

**2.2 Use Case-Models**

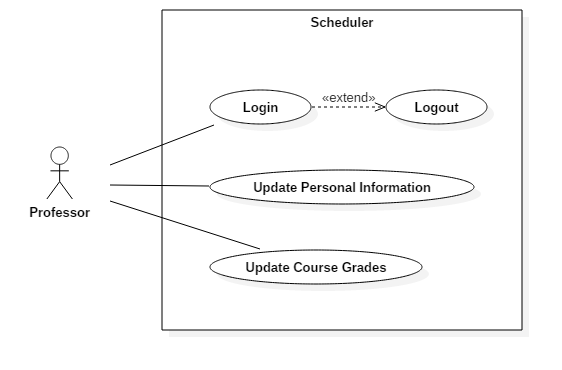
**System Administrator:**



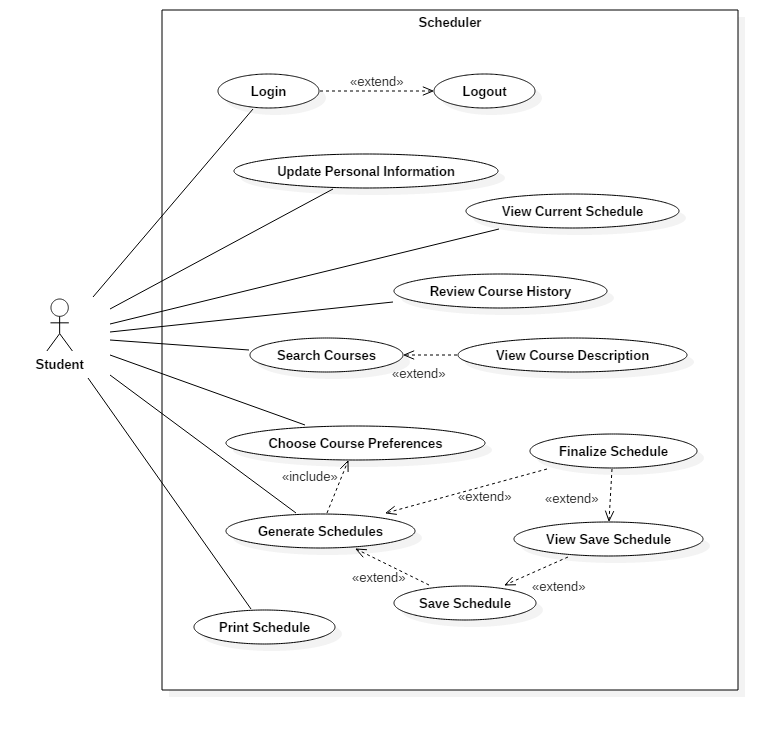
**Faculty Administrator:**



**Professor:**

****

**Student:**

****

**Use Case-Description**

|  |  |
| --- | --- |
| UC1 - Login | |
| Risk Assessment | 5/5 |
| Importance | 5/5 |
| Actor(s) | Student, Teacher, 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 |

|  |  |
| --- | --- |
| UC2  - Logout | |
| Risk Assessment | 5/5 |
| Importance | 3/5 |
| Actors | Student, Teacher, 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. |

|  |  |
| --- | --- |
| UC# - Update Personal Information | |
| Risk Assessment | 5/5 |
| Importance | 5/5 |
| Actors | Student, Teacher, 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 |

|  |  |
| --- | --- |
| UC# - Review Course History | |
| Risk Assessment | 2/5 |
| Importance | 4/5 |
| Actors | Student |
| Description | Review student’s course history which includes the semester 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 |

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

|  |  |
| --- | --- |
| UC# - Choose Course Preferences | |
| Risk Assessment | 2/5 |
| Importance | 3/5 |
| Actors | Student, Faculty Admin |
| 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 to student’s account |
| Post-condition | Success: User preferences are saved  Fail: User preferences are not saved |

|  |  |
| --- | --- |
| UC# -  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 |

|  |  |
| --- | --- |
| UC# - 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 |

|  |  |
| --- | --- |
| UC# - 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 |

|  |  |
| --- | --- |
| UC# - View Saved Schedules | |
| Risk Assessment | 1/5 |
| Importance | 4/5 |
| Actors | Student |
| Description | User can view the different schedules offered to him or her |
| Basic Flow | 1. User requests to view his/her saved schedules 2. System displays user’s saved schedules |
| 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 |

|  |  |
| --- | --- |
| UC# - Print Schedule | |
| Risk Assessment | 1/5 |
| Importance | 1/5 |
| Actors | Student |
| Description | User prints his/her finalized 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 |

|  |  |
| --- | --- |
| UC# - Update Course Grades | |
| Risk Assessment | 4/5 |
| Importance | 5/5 |
| Actors | Teacher |
| 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 teacher of the selected course |
| Post-condition | Success: Students’ grades in the selected class are updated  Fail: Students’ grades are not updated |

|  |  |
| --- | --- |
| UC# - 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 |

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

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

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

|  |  |
| --- | --- |
| UC# - View Current Schedule | |
| Risk Assessment | 1/5 |
| Importance | 5/5 |
| Actors | Student |
| Description | Student sees their current semester schedule |
| Basic Flow | 1. Student logs in 2. Student views current schedule |
| Pre-condition | Student is logged in |
| Post-condition | Success: Student sees current schedule  Fail: Student does not see a current schedule |

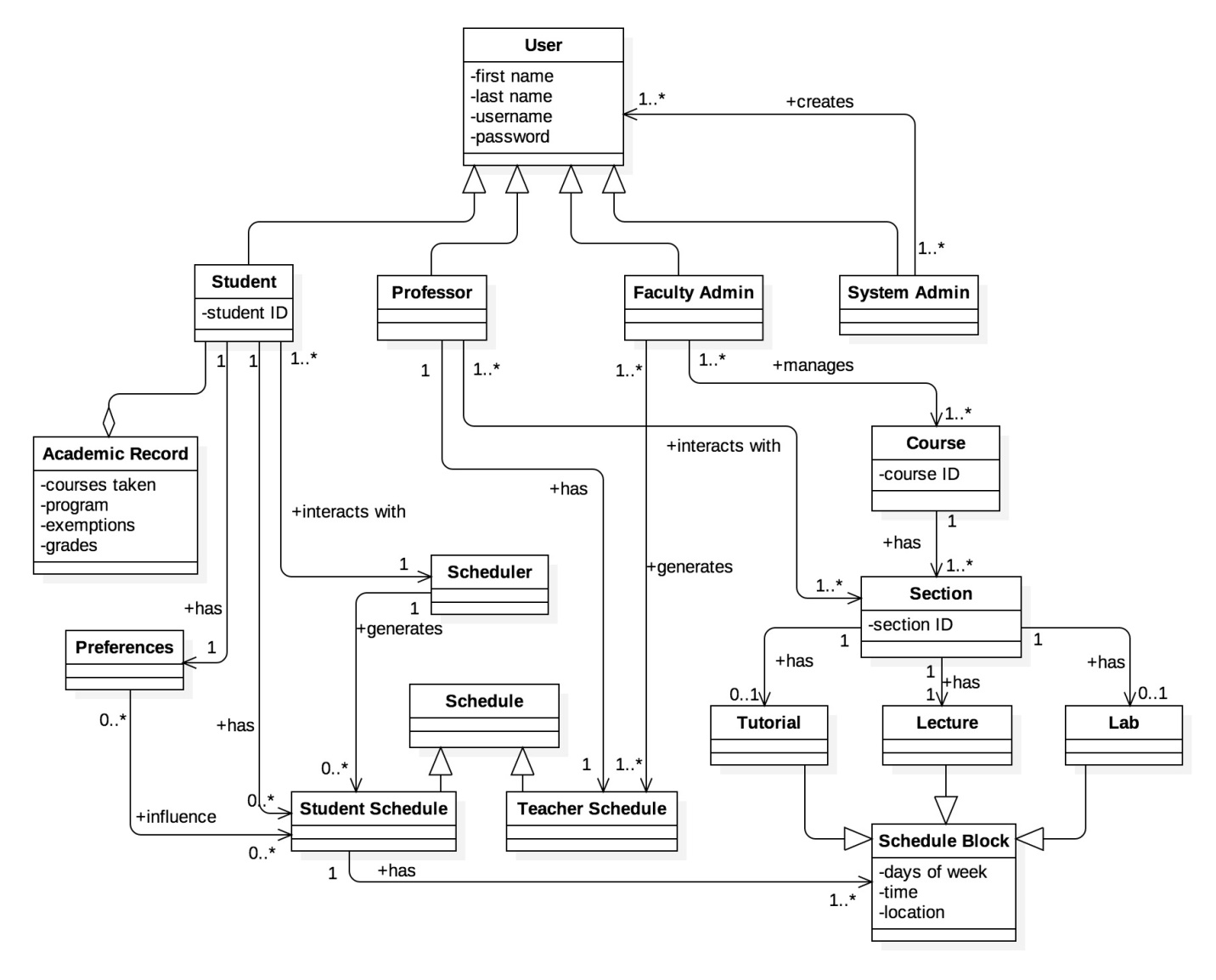
|  |  |
| --- | --- |
| UC# - View Saved Schedules | |
| Risk Assessment | 1/5 |
| Importance | 3/5 |
| Actors | Student |
| Description | Student views previously saved schedules that were generated from scheduler. |
| Basic Flow | 1. Student logs in 2. Student runs scheduler 3. Student saves schedule(s) 4. Student views saved schedule(s) |
| Pre-condition | Student is logged in |
| Post-condition | Success: Student sees saved schedules  Fail: Student does not see saved schedule page |

|  |  |
| --- | --- |
| UC# - Create Course | |
| Risk Assessment | 3/5 |
| Importance | 5/5 |
| Actors | Faculty Admin |
| Description | Faculty Admin creates a new course. |
| Basic Flow | 1. Faculty Admin is logged in 2. Faculty Admin creates course 3. Faculty Admin enters course information 4. Course is created |
| 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. |

|  |  |
| --- | --- |
| UC# - Remove Course | |
| Risk Assessment | 5/5 |
| Importance | 4/5 |
| Actors | Faculty Admin |
| Description | Faculty Admin removes a course. |
| Basic Flow | 1. Faculty Admin logs in 2. Faculty Admin searches a course 3. Faculty Admin finds course to remove 4. Faculty Admin 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 |

|  |  |
| --- | --- |
| UC# - 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. |

**2.3 Domain Model**

****

**2.4 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.

* 1. Security
     1. The access to the system will be granted only by providing correct credentials (i.e. username and password)
     2. The password must have a length between 8 to 16 characters alphanumeric characters.
     3. The communication protocol must be encrypted (i.e. the system will be access by HTTPS)
     4. The user’s password will be stored encrypted in the database, using SHA-2
     5. After 10 minutes of inactivity, the system shall automatically logout the user.
     6. The system must provide the options to manage users and permissions.
  2. Interoperability
     1. The system must have the option to download the information in PDF and CSV format.
     2. The system must have the option to export the schedule to Outlook and Google calendars.
     3. The system must be have the option to be integrated (embedded) to the current University portal, if required (SSO)
     4. The system must have the option to be integrated to standard directory protocol (LDAP)
  3. Normative
     1. The system’s information must be archived for a period of 4 years.
     2. The system must complain the current Quebec normative for educational web applications.

1. **Reliability**
   1. Availability
      1. 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)
   2. Recoverability
      1. On case of system failure, the maximum restore time must be 4 hours.
      2. On case of database failure, the maximum information lost must be 1 day.
2. **Usability**
   1. Understandability
      1. 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)
      2. The system must be offered in English and French. I must have to option to change its language.
   2. Accessibility
      1. To grant access to senior people, the minimum system’s font size will be 14px, along with soft, no high contrast colors.
      2. To grant access to blind people, the system should have menu voice assistance and screen readers (visually impaired user's technology)
3. **Efficiency**
   1. Time
      1. The system’s response time will be less than 3 seconds in all operations.
   2. Resources
      1. The system must designed to run on medium size server (4-8 Cores, 16-32 GB RAM)
4. **Maintainability**
   1. Scalability
      1. The system must be able to support 5.000 users, and allow a grow up to 20.000 users.
      2. The system must be able to serve 500 concurrent users.
   2. Support
      1. To help support, technical documentation about the system use, design, installation, configuration and troubleshooting must be created, and the source code must be documented.
   3. Testability
      1. Evidence about the test cases performed during its development must be provided.
5. **Portability**
   1. Adaptability
      1. The system must adapt its interface for both laptop/desktop browser and mobile devices with screens 8 inch or bigger.
      2. The system must work on the latest versions of Chrome and Firefox.
      3. The system must be build using modern web standards (HTML5, CSS3).

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

**3.RESOURCE EVALUATION**

In this section, the resources available in the building of the Apollo application will be provided. Human resources, in terms of each members’ 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.

**3.1 Human Resources**

**Francis Bouchard:**

Capabilities:

* Took SOEN 287 (Web programming class) where we learned various languages such as HTML, CSS, Javascript and PHP.
* 4 months of co-op internship as a web developer at UQAM. Developed RESTful APIs to replace legacy applications.
* Experience in Node.js and Angular.
* 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.

**Liuai Hatter:**

Capabilities:

* SOEN 287: But would say I only have basic knowledge of PHP.
* ENGR 301(Eng. Management Principles and Economics): MS Project, work breakdown schedule, Gantt charts.
* Bachelor of Fine Arts and worked as a graphic designer (all for web) for a tech company.

Availability:

Est. 5 hours / week (meetings included).

Available to meet weekdays and evenings, sometimes weekends.

**Wahab Ahmed:**

Capabilities:

* Took SOEN 287 a year ago. We learnt HTML, PHP, JS and briefly touched databases for the final website assignment.
* Worked as a mechanical CAD drafter at some Toronto based company.

Availability:

I can contribute 5 hours to the project weekly, meetings included. Available to meet any time that doesn't clash with my class schedule with weekends included.

**Philippe Abou Kasm:**

Capabilities:

* COEN 243/244: Knowledge in object oriented programming in C++.
* COEN 352: Knowledge in data structures and algorithms and familiarity with Java programming.
* COEN 311: Basic knowledge in assembly language programming.
* Basic knowledge in HTML.

Availability:

Willing to contribute 5 hours per week meeting included. Available sometimes on weekends.

**Jian Huang:**

Capabilities:

* SOEN 287: learnt HTML5, CSS, Javascript and a bit of PHP
* ENGR 301: Took management course for engineering

Availability:

I can contribute to as many hours (5 or more hours) as possible to the project as I only have assignments to do.

**Ricardo Cortes**

Capabilities:

* Electronic/Computer Engineer (Undergraduate), M. Eng. Software Engineering (current)
* SOEN 6471 Advanced Software Architectures
* SOEN 6481 System Requirement Specification
* SOEN 6461 Software Design Methodologies

Availability :

Monday to Friday only. Estimated Time: 4 hours per week, plus meetings

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. Technical strength: Java, Design Patterns, Architectural Patterns.

**Sabrina Ashraff:**

Capabilities:

* SOEN 287: Learned HTML5, CSS, Javascript and PHP
* 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:

EST 4 hours/week (meetings included). 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.

**Clozzy-Mailey Chavez**

Capabilities:

* COEN 243/244/352: Learned object oriented programming in C++ and a little bit of Java.
* ENGR 290/301: Learned about project management, technical writing and technical presentation in a team environment.
* Basic knowledge in HTML.

Availability:

Est. 5 hours/week, meetings included.

**Anna Rogozin**

Capabilities:

* 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:

5 hours/week (meetings included), prefer in the evening or during the weekend.

**Ramy Sandouk:**

Capabilities:

* Second year computer engineering
* Coen 243, 244, 352, 311

Availability:

Can contribute 4 to 5 hours per week.

**Matthew Teolis:**

* 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.
* Strengths: Good/efficient coder

Availability:

5 hours/week, meetings included, minimum.

**3.2Technical 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

**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:

**2. System Administrator**

    2.1. System administrator can create new users

        2.1.1. System administrator can grant new users specific permissions

    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

**Reason:** Unable to implement due to time constraints

**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

**Reason:** Unable to implement due to time constraints

**4. Professor**

    4.1. Professor can view his or her course schedule

    4.2. Professor can update students’ grades

**Reason:** Unnecessary for general functionality of the system

**5. Student**

5.4. Student can generate schedules

        5.4.4. Student can finalize a schedule

**Reason:** Unnecessary for general functionality of the system