# 2 - Project Description

Concordia University's current course enrolment system does not provide students with all their possible scheduling options for their respective courses. In a program such as Software Engineering where sections are limited and professors must be engineers, it can prove difficult for a student to create a desirable schedule. The Force is a web based application that allows students to choose from a list of generated course schedules based on their desired courses.

The Force has a simple interface where users are prompted to enter their desired courses along with certain criteria, such as requesting to have no classes on Mondays or to not finish later than 5PM. Next, all schedule options that satisfy the inputted information are displayed. The student is first verified by the system for eligibility in the indicated courses, based on prerequisites, co-requisites, and program restrictions. Users can also choose to add and remove courses manually from their schedules.

This project is documented through five deliverables, each of which describes progress in the creation of this application:

* Deliverable 1: Overview and general system description
* Deliverable 2: Project requirements and plan of action
* Deliverable 3: Description of system architecture and design
* Deliverable 4: Implementation and testing of refined system prototype
* Deliverable 5: Concluding system implementation and final report

# 3 - Goals and Constraints

# **3.1 - Functional Requirements**

1. Interface - Login Page

1. Display brief message describing function of application
2. Display fields for login credentials
3. Login button leads to section 2

2. System - Login Page

1. Verify credentials from login page
   1. If credentials are valid, proceed to section 3
   2. If credentials are invalid, return to section 1 and display message for invalid information

3. Interface - Course Page

1. List of saved schedules (initially empty)
2. List of desired courses (initially empty)
3. Button for adding course
   1. Prompt user for course code
   2. List all sections as checklist for user to select from
   3. When course is added, add to schedule
4. Button for removing course next to each course in schedule
5. Button for generating schedules based on predetermined Concordia Course Sequence
6. Additional criteria section for requesting days off and requesting earliest/latest course times

4. System - Course Page

1. Verify student eligibility for added course
   1. Check course prerequisites and verify if student has them completed
   2. Check course co-requisites and prompt the user to add the co-requisite course to list automatically.
   3. Check program restrictions (such as necessary credits) for course
   4. Verify course vacancy for added course
2. If all conditions are met, add course to schedule
3. If any of these conditions are not met, display message to user explaining why they cannot enroll in this course
4. If at least one course schedule can be generated, proceed to section 5. Otherwise, display message informing user that there are no possible schedules with the selected courses and criteria.

5. Interface - Course Scheduling Page

1. Display generated schedules based on course sequence and criteria
2. Below each generated schedule:
   1. List sections used for respective schedule
   2. Button for downloading schedule as PDF
   3. Button for adding schedule to list of saved schedules

6. System - Course Scheduling Page

1. If user adds schedule to saved schedules, store generated schedule in profile for future logins

7. System - Logout

1. If user logs out using logout button, save session information and logout user
2. If user closes window, save session information and logout user
3. If user is inactive for five minutes, save session and information and logout user

8. Interface - Reset Password

1. Display field for account email
2. Reset button sends password reset email

9. System - Reset Password

1. Verify if account email is valid
2. If valid, send 6-character temporary password to email address
3. If invalid, display message saying that account does not exist

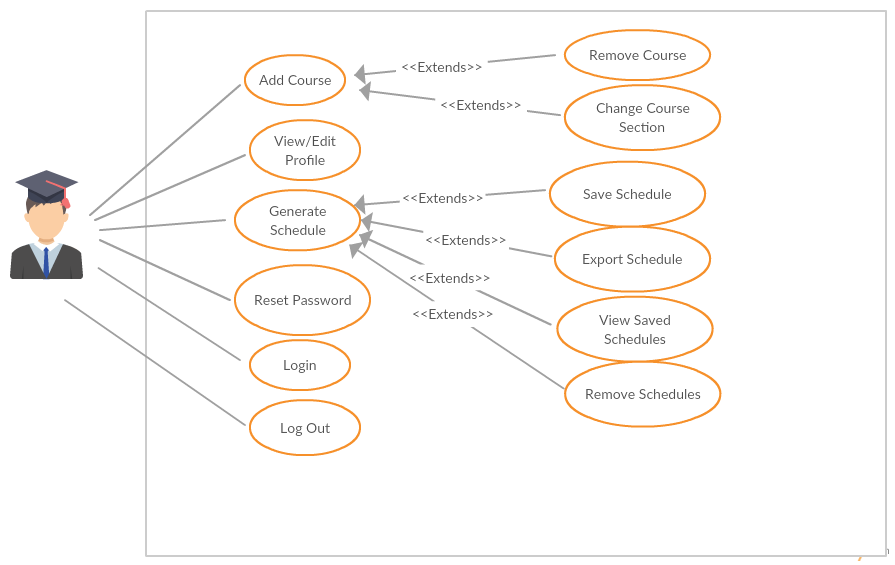
10. Interface - View profile

1. Can be viewed anytime after logging in
2. Display student record, including program, credits, courses taken, and general student information
3. Button for changing password

11. System - View profile

1. Retrieve student record from Concordia system
2. Change stored password if user changes password
3. Change stored email if user changes email

**Use Case Diagram**



|  |  |
| --- | --- |
| ID: | UC01 |
| Name: | Login |
| Importance (/5): | 5/5 |
| Difficulty (/5): | 2/5 |
| Risk Assessment: | Low |
| Actors: | Student |
| Goals: | * To allow students to access the main site’s home page |
| Summary: | By using this feature, the student will be able to view their profile information, including ID number, first name, last name, program, starting semester. They will also have a link to their student transcript. |
| Preconditions: | The user must be on the login page |
| Basic Flow: | 1. The user must provide valid username and password 2. The user is redirected to the main page |
| Post Conditions: | Success: The user will be brought to the main site’s home page  Failure: The user will remain on the login page since he/she will be denied access. An error message will be displayed. If the user cannot figure out their password, they must go through the proper procedure to change it. |

|  |  |
| --- | --- |
| ID: | UC02 |
| Name: | Logout |
| Importance (/5): | 5/5 |
| Difficulty (/5): | 2/5 |
| Risk Assessment: | Low |
| Actors: | Student |
| Goals: | * To allow students to log off of their account in the system |
| Summary: | Any user who can log out of the system |
| Preconditions: | User must have successfully logged into the system |
| Basic Flow: | 1. User selects “Log Out” 2. System prompts the user to confirm the log out 3. User is brought back to the main page |
| Post Conditions: | Success: User is brought back to the main page  Failure: User remains on the same page and an error is displayed |

|  |  |
| --- | --- |
| ID: | UC03 |
| Name: | View/ Edit Profile |
| Importance (/5): | 5/5 |
| Difficulty (/5): | 3/5 |
| Risk Assessment: | Low |
| Actors: | Student |
| Goals: | * To allow students to view/edit their profile information |
| Summary: | By using this feature, the student will be able to view their profile information, including ID number, first name, last name, program, starting semester, change password and change email. They will also have a link to their student transcript. |
| Preconditions: | User must have successfully logged into the system |
| Basic Flow: | 1. User clicks on their name to view their Profile 2. System displays the user’s profile 3. The user can choose to edit any information |
| Post Conditions: | Success: User returns to main, changes are saved  Failure: User remains on current pages, changes unsaved |

|  |  |
| --- | --- |
| ID: | UC04 |
| Name: | Reset Password |
| Importance (/5): | 5/5 |
| Difficulty (/5): | 3/5 |
| Risk Assessment: | High |
| Actors: | Student |
| Goals: | * To allow students to reset their password if forgotten |
| Summary: | The user can reset his/her password by providing their email address. An email will be automatically sent to the email address with instructions on how to reset the password. |
| Preconditions: | The user must be on the login page |
| Basic Flow: | 1. User clicks on the reset password option 2. System asks the user for their username |
| Post Conditions: | Success: Email is sent to the user, user remains on current page  Failure: Email is not sent, user remains on current page |

|  |  |
| --- | --- |
| ID: | UC05 |
| Name: | Add Course |
| Importance (/5): | 5/5 |
| Difficulty (/5): | 3/5 |
| Risk Assessment: | Medium |
| Actors: | Student |
| Goals: | * To allow students to add specified course to his/her schedule |
| Summary: | Specified courses can be successfully added by the user to his/her schedule. |
| Preconditions: | * The user must have logged in & be on the main page * The system has access to the course list & schedules * The system has access to previously created schedule * The user must have the pre-requisites (System will check) |
| Basic Flow: | 1. User clicks on add course option 2. User searches for the class they would like to add 3. User confirms the course they would like to add 4. System indicates if the add was successful after checking the pre-requisites/co-requisites. |
| Post Conditions: | Success: Specified course is added to the schedule  Failure: Process fails, Error is displayed to the user |
| Notes | \*\*\*\*\*Registration constraints include the following:   * Timeslots for courses cannot overlap (minimum 15 minutes between back-to-back courses). * Students cannot register for the same course twice within the same semester. * Students cannot register for a course which they already completed. * Students cannot register for courses for which they have not fulfilled the pre-requisites/co-requisites |

|  |  |
| --- | --- |
| ID: | UC06 |
| Name: | Change Course Section |
| Importance (/5): | 5/5 |
| Difficulty (/5): | 3/5 |
| Risk Assessment: | Medium |
| Actors: | Student |
| Goals: | * To allow students to change the section of their course |
| Summary: | Specified courses can change their sections in the user’s schedule. |
| Preconditions: | * The user must have logged in & be on the main page * The system has access to the course list & schedules * The system has access to previously created schedule * The Course must already be on the schedule |
| Basic Flow: | 1. User clicks on specified course 2. User chooses from other available sections of the course 3. System responds with a change in the schedule |
| Post Conditions: | Success: Specified course section is changed  Failure: Process fails, Error is displayed to the user |

|  |  |
| --- | --- |
| ID: | UC07 |
| Name: | Remove Course |
| Importance (/5): | 5/5 |
| Difficulty (/5): | 3/5 |
| Risk Assessment: | Medium |
| Actors: | Student |
| Goals: | * To allow students to remove courses from their schedule |
| Summary: | Specified courses can be removed by the user from his/her schedule. |
| Preconditions: | * The user must have logged in & be on the main page * The system has access to the course list & schedules * The system has access to previously created schedule * The Course must already be on the schedule |
| Basic Flow: | 1. User clicks on specified course 2. System responds by removing the course in the schedule |
| Post Conditions: | Success: Specified course is removed  Failure: Process fails, Error is displayed to the user |

|  |  |
| --- | --- |
| ID: | UC08 |
| Name: | Generate Schedule |
| Importance (/5): | 5/5 |
| Difficulty (/5): | 5/5 |
| Risk Assessment: | High |
| Actors: | Student |
| Goals: | * To allow students to auto-generate a schedule through the system |
| Summary: | The system will auto-generate a schedule for the user |
| Preconditions: | * The user must have logged in & be on the main page * The system has access to the course list & schedules * System indicates if the add was successful after checking the pre-requisites/co-requisites. |
| Basic Flow: | 1. User clicks on Generate course 2. System responds by creating a schedule |
| Post Conditions: | Success: Schedule is auto-generated  Failure: Process fails, Error is displayed to the user |
| Notes | \*\*\*\*\*Registration constraints include the following:   * Timeslots for courses cannot overlap (minimum 15 minutes between back-to-back courses). * Students cannot register for the same course twice within the same semester. * Students cannot register for a course which they already completed. * Students cannot register for courses for which they have not fulfilled the pre-requisites/co-requisites |

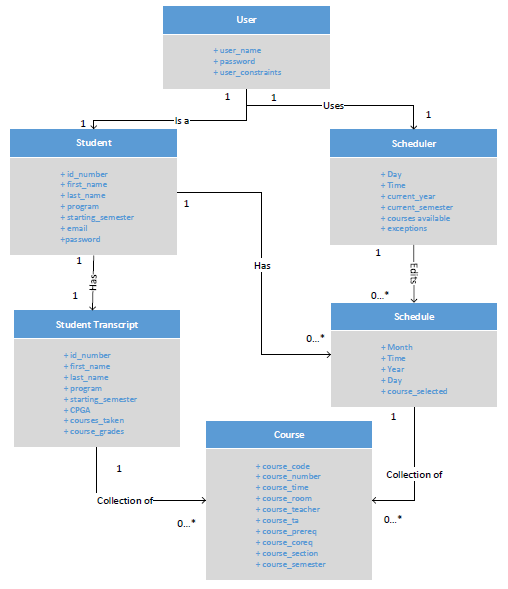
|  |  |
| --- | --- |
| ID: | UC09 |
| Name: | Export Schedule |
| Importance (/5): | 3/5 |
| Difficulty (/5): | 5/5 |
| Risk Assessment: | Medium |
| Actors: | Student |
| Goals: | * To allow students to export their schedules and save it on their personal computers |
| Summary: | Students can choose to export their schedules to PDF or as an image and save it to their computer |
| Preconditions: | * The user must have logged in & be on the main page * The user must have a schedule completed |
| Basic Flow: | 1. User clicks on their saved schedule 2. System displays the selected schedule 3. User clicks on export option 4. System prompts for the file type and save location |
| Post Conditions: | Success: Schedule is exported and saved to proper location  Failure: Process fails, Error is displayed to the user |

|  |  |
| --- | --- |
| ID: | UC10 |
| Name: | Save Schedule |
| Importance (/5): | 3/5 |
| Difficulty (/5): | 3/5 |
| Risk Assessment: | Medium |
| Actors: | Student |
| Goals: | * To allow students to save their schedules for future access |
| Summary: | The schedules generated by the system and edited by the user can be saved as a preference |
| Preconditions: | * The user must have logged in & be on the main page * The user must have a schedule completed |
| Basic Flow: | 1. While on the generated schedule, the user clicks on save schedule 2. System prompts the user for confirmation, then saves the schedule. |
| Post Conditions: | Success: Schedule is saved and added to preference list  Failure: Process fails, Error is displayed to the user |

|  |  |
| --- | --- |
| ID: | UC11 |
| Name: | View Saved Schedule |
| Importance (/5): | 2/5 |
| Difficulty (/5): | 3/5 |
| Risk Assessment: | Medium |
| Actors: | Student |
| Goals: | * To allow students to view their saved schedules |
| Summary: | The user can load any schedule previously created and stored as a preference. |
| Preconditions: | * The user must have logged in & be on the main page * The user must have a schedule in the preferred section * The system must have access to all saved schedules |
| Basic Flow: | 1. User clicks on “My Saved Schedules” 2. System provides the list of the schedules saved, waiting for the user to select the desired one 3. Once selected by the user, the system displays the schedule |
| Post Conditions: | Success: Schedule is displayed from the preference list  Failure: Process fails, Error is displayed to the user |

|  |  |
| --- | --- |
| ID: | UC12 |
| Name: | Remove Schedule |
| Importance (/5): | 1/5 |
| Difficulty (/5): | 3/5 |
| Risk Assessment: | Medium |
| Actors: | Student |
| Goals: | * To allow students to remove a saved schedules |
| Summary: | The user can remove any schedule they no longer want from the list of preferred schedules. |
| Preconditions: | * The user must have logged in & be on the main page * The user must have a schedule in the preferred section * The system must have access to all saved schedules |
| Basic Flow: | 1. User clicks on “My Saved Schedules” 2. System provides the list of the schedules saved, waiting for the user to select the desired one 3. Once selected by the user, the system removes the schedule |
| Post Conditions: | Success: Schedule is removes from the preference list  Failure: Process fails, Error is displayed to the user |

# **3.2 - Domain Model**



User

* The ‘User’ class is an entity representing a user of the system. This class contains 3 attributes. They are user\_name, password and user\_constraints. The username and password are used for logging into the website. The user\_constraints are limitations entered by the user such as number of days off, time free and courses to avoid.
* The ‘User’ class is related to one ‘Student’ class to represent that each user is a registered student at Concordia University in the 174
* The ‘User’ class is also related to one ‘Scheduler’ class to represent that each user will use the scheduler tool to create schedules in the system.

Scheduler

* The ‘Scheduler’ class is an entity representing the key concept of the system, the schedule creation. The class contains 6 attributes; Day, Time, current\_year, current\_semester, courses\_available and exceptions. Day, Time, current\_year and current\_semester and variables that will be filled when a student add/drops a class in their schedule. Courses\_available are courses that will be given in a certain semester. Exceptions will be created on the user\_constraints entered by the user.
* The ‘Scheduler’ class is related to 0 or more ‘Schedule’ objects to represent that the scheduler can create one or more schedules for various semesters and years

Schedule

* The ‘Schedule’ class is an entity representing the schedule made by a user for a certain semester/year. The class contains 5 attributes. They are Month, Time, Year, Day and course\_selected (All self-explanatory).
* The ‘Schedule’ class is a subset of the scheduler is related to 0 or more ‘Courses’ to represent that a ‘Schedule’ is a collection of courses made by a ‘Student’.

Student

* The ‘Student’ class is a specialized case of the user. The class contains 5 attributes; id\_number, first\_name, last\_name, program, starting\_semester. Email and password. (All self-explanatory).
* Each 'Student' in the system is associated one 'Student Transcript'. This is the student’s academic history. Also, each ‘Student’ is related with 0 or more ‘Schedule’ which represent the user’s schedules for any academic year.

Student Transcript

* The ‘Student Transcript’ class is an entity representing courses that the ‘Student’ has already taken. This also shows other information about the ‘Student’.
* Each ‘Student Transcript’ is related to 0 ore more ‘Courses’ to represent the classes that a ‘Student has already taken’.

Course

* The ‘Course’ class is an entity detailing all the information about courses available to software engineering students during various years/semesters.

# **3.3 - Constraints and Qualities**

Documentation: In order to complete the project in an effective manner, all planning and progress must be documented in an organized manner. This documentation includes both technical and non-technical specifications.

Performance: The generator shall quickly generate the schedules upon pressing the generate button (>10 seconds). Adding, removing, and modifying courses from course list is done instantly.

Usability: Simple, minimalistic interface without clutter. User must know what courses they want to enroll in for generator to be effective. Generator cannot be used

to enroll in courses, but instead to guide students to create an optimal schedule with the course sections.

Reliability: The generator accurately displays all possible schedule options. Issues can

arise where class times are changed, but the generator is synchronized with

the Concordia system for instant changes. Verifying prerequisites and course restrictions is completed using the student's record for accurate results.

Scalability: The generator currently works for Software Engineering students and will

have records for all relevant courses in the program. Additional courses could be added if needed and the application could be modified to apply to other programs as well.

Security: The application could pose some security concerns, but since the tool cannot be used to modify any student records, they are not very significant. The main security risk involves the storing of account passwords, which will be encrypted into hashcode to prevent data mining. Additionally, the the student records should only be available for viewing by the respective user after logging in.

# 4 – Resource Evaluation

# **4.1 – Human Resources**

|  |  |
| --- | --- |
| Name: | Stefano Pace |
| Student ID: | 27454716 |
| Sub Team: | Documentation (Sub-Team Leader)/Coordination |
| Knowledge: | Java, HTML, CSS, JavaScript, PHP, XML, SQL, Ajax, Databases, Prolog, Lisp, Ruby, Common Lisp Object System, Visual Basic, Python, Microsoft Office: Word, Excel, PPT, Outlook, Publisher, Eclipse, Notepadd++, Omnicast 4.6 |
| Experience: | 1. IT Telecom Integration 2. Wide Area Network (WAN) design 3. Analyst End User Support |
| Strengths: | * Teamwork * Time Management * Organization * Leadership |
| Time for Project: | 5 hours/week |

|  |  |
| --- | --- |
| Name: | Julian Ippolito |
| Student ID: | 27419112 |
| Sub Team: | Front-End Programming (Sub-Team Leader) |
| Knowledge: | Java, C#, JavaScript, HTML5, CSS, XML, PHP, MySQL, Ruby |
| Experience: | 1. IT Manufacturing Applications 2. Programming in C# |
| Strengths: | * Teamwork * Time Management * Efficiency |
| Time for Project: | 5 hours/week |

|  |  |
| --- | --- |
| Name: | Adam Arcaro |
| Student ID: | 27459157 |
| Sub Team: | Documentation |
| Knowledge: | Java, C#, JavaScript, HTML5, CSS, XML, PHP, MySQL, Ruby |
| Experience: | 1. IT Application Specialist 2. Data Entry for Contract Management 3. Customer Service |
| Strengths: | * Teamwork * Can handle Pressure Situations * Strategy |
| Time for Project: | 5 hours/week |

|  |  |
| --- | --- |
| Name: | Joey Tedeschi |
| Student ID: | 27513062 |
| Sub Team: | Documentation |
| Knowledge: | Java, Ruby (beginner), Python (beginner), PHP, JavaScript, JQuery, HTML, CSS, MYSQL |
| Experience: | 1. Web Development 2. Software Development |
| Strengths: | * Teamwork * Organization * Object Oriented Programming |
| Time for Project: | 5 hours/week |

|  |  |
| --- | --- |
| Name: | Olivier Cameron-Chevrier |
| Student ID: | 27228805 |
| Sub Team: | Server-side Programming |
| Knowledge: | Java, JavaScript, PHP, Html, CSS, SQL, Ruby, Prolog, CLOS |
| Experience: | 1. Customer Service/Support 2. Web Programming |
| Strengths: | * Teamwork * Time Management |
| Time for Project: | 5 hours/week |

|  |  |
| --- | --- |
| Name: | Georges Mathieu |
| Student ID: | 26863477 |
| Sub Team: | Server-side Programming (Sub-Team Leader) |
| Knowledge: | C, C++, Java, C#, VisualBasic, HTML, CSS, Javascript, JQuery, PHP, JSP, COBOL, AS/400, Prolog, LISP |
| Experience: | 1. Junior Data Architecture 2. Web Programming |
| Strengths: | * Organization * Time Management * Strategy |
| Time for Project: | 5 hours/week |

|  |  |
| --- | --- |
| Name: | Hasan Ahmed |
| Student ID: | 27546467 |
| Sub Team: | Front-End Programming |
| Knowledge: | Java, Python (Basic), HTML and CSS, PHP & MySQL |
| Experience: | 1. Customer Service 2. Web Programming |
| Strengths: | * Organization * Time Management * Creative Solutions |
| Time for Project: | 5 hours/week |

|  |  |
| --- | --- |
| Name: | Marc-Andre Dragon |
| Student ID: | 27721579 |
| Sub Team: | Server-side Programming |
| Knowledge: | C# (beginner), Java (intermediate), Python (beginner), PHP (beginner), JavaScript (beginner), Html (beginner), CSS (beginner) |
| Experience: | 1. Platforms (Gaming) 2. Web Programming |
| Strengths: | * Organization * Time Management * Critical Thinking |
| Time for Project: | 5 hours/week |

|  |  |
| --- | --- |
| Name: | Jordan Stern |
| Student ID: | 27400144 |
| Sub Team: | Front - side Programming |
| Knowledge: | Java, Javascript, HTML5, CSS, AJAX, Prolog, Common Lisp, Adobe Creative Suite |
| Experience: | 1. Customer Service 2. Web Programming |
| Strengths: | * Organization * Programming * Team Work |
| Time for Project: | 5 hours/week |

|  |  |
| --- | --- |
| Name: | George Theophanous |
| Student ID: | 27400578 |
| Sub Team: | Quality Assurance (Team Leader) |
| Knowledge: | Java, VBA, Python, LISP, Ruby, LISP Common Object, Assembly, HTML, Prolog, C# |
| Experience: | 1. QA/Software Development 2. Web Programming |
| Strengths: | * Organization * Programming * Critical Thinking |
| Time for Project: | 5 hours/week |

|  |  |
| --- | --- |
| Name: | Kuan Jiang |
| Student ID: | 27352824 |
| Sub Team: | Documentation (Sub-Team Leader)/Coordination |
| Knowledge: | Java, JavaScript, PHP, SQL, HTML, CSS, Python, Apache, AJAX, UML, Ruby, C++, C# |
| Experience: | 1. QA/Software Tester 2. Auditing |
| Strengths: | * Debugging * Programming |
| Time for Project: | 5 hours/week |

# **4.2 – Technical Resources**

To facilitate communication between developers, Skype, Slack and GitHub are used. Skype is a service provided by Microsoft that allows the use of webcams, microphones and screen sharing for instant messaging and online voice/video conferences.

All files used for this project were sorted and stored using GitHub. All developers were given access to the repository and its content. To make sure that the source code used was maintained properly, the code files were given version numbers. Since Slack has GitHub Integration, it was also used for instant messaging and served as a way to update all developers of changes to the GitHub repository.

The server used to host our system, *The Force*, has the following hardware specifications:

* Windows Server 2012 Operating System (64-bit)
* Octa-Core Processor (i7-4790 CPU @ 3.60GHz)
* 16 GB DDR2 RAM
* 25 Mbps Bandwidth
* 2TB Memory

The Software used on the server will be the following:

* HTML, CSS, PHP, MySQL
* MySQL workbench
* GitHub desktop
* NotePad++
* Eclipse
* Bootstrap

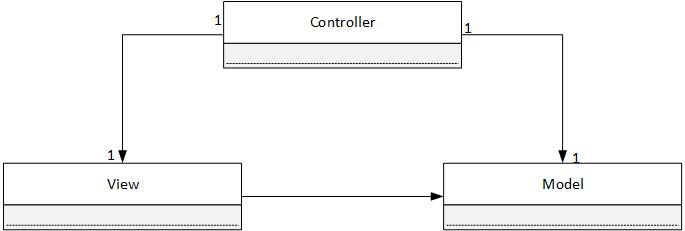
# 5 - Scoping

All features mentioned in previous sections will be implemented into the system. It was determined that, due to a lack of time and complexity in implementation, the following non-essential features would deemed out of scope and removed:

|  |  |
| --- | --- |
| Feature Removed | Reason |
| User Account Deactivated After 5 Failed Attempted Logins | This requirement was scoped out due to time constraints |
| Generate course schedule for selection with alternate professor | This can be done by the “changing section” use case, as the section information will include the name of the professor in its information |
| Generate course schedule for user defined course code & sequence | This requirement was considered to be not widely desired as Concordia University suggests all students follow the pre-determine sequence. As well, this feature proved to be too complex to implement and was scoped out due to time constraints. |
| Website Timeout | This requirement was scoped out due to time constraints |
| Manual Registration | The team realized that all students registered at Concordia University would already have an account, so manual registration was not required. |

# Solution Sketch

## **Architecture**



The Force will be following a Model-View-Controller (MVC) software architecture pattern as its high-level architecture. The MVC architecture is ideal for this project as it allows for a clear separation between domain logic, controllers/requests, and user interface into unique classes. By isolating these key components, the MVC architecture enables development and testing of all components without requiring them to interact and possibly interfere with each other. In a team project such as this, independence is key. By allowing the controller, model, and view of a page to be modified simultaneously without worry of possibly affecting other components of the application, greater progress can be made for each portion by the various sub-teams.

In the case of The Force, as it is a web-based application, any request the user makes on a given web page (the view) is handled by the controller. Based on the inputs of said user, the controller then makes a call to the model in order to determine the proper behavior in which the data needs to be treated, and then applies said logic using the domain objects. In most cases, requests for state-specific information are usually presented by the view, while instructions to update states are given by the controller. Once the model has handled the logic required to gather the data necessary for the user’s request, the web page (or view) will be updated using the data acquired by the controller. The end result is a new web page, displayed on the user’s browser, on which the cycle can then repeat as required.

In comparison to MVC structures, Web Enterprise Architecture (WEA) is composed of layers instead of simply components. Separation of concerns is still present in WEA, but high-level layers may need to interact with lower-level layers instead of just those directly below it, increasing the complexity of the task and performance time as a whole. It would also be far more difficult to create clean separations between layers in comparison to simply separating all aspects into 3 classes. It is for these reasons that the Force chose to implement MVC as it’s high-level architecture: The separation of concerns it provides, and the ease at which this separation can be done.

## **Technologies in Use**

* + 1. **Programming Languages:**

1. *HTML*

Hyper Text Markup Language (HTML) is a markup language used to describe web documents, and document content. Seeing as HTML is the standard language read by web browsers, and The Force is a web-based application, it stands that HTML would be a requirement for developing the user interface.

1. *CSS*

Cascading Style Sheets (CSS) is a stylesheet language used to describe the layout and appearance of a document written in a markup language (in the case of this application, HTML). CSS is imperative to the development of The Force as it is a simple way to not only to orient information on screen for the user, but do so in an easily reusable manner.

1. *Javascript*

Javascript is a client-side scripting language, considered to be the standard client-side scripting language for web applications. As all modern web browsers support it, Javascript is ideal for development purposes and will be the main language used for client-side programming. The fact that it is a multi-paradigm scripting language also makes it incredibly flexible in use. The inclusion of Javscript not only provides The Force with a complete GUI, but its quick learning curve makes it perfect for those unfamiliar with the language. Javascript will be used in order to design how the various web pages behave based on event occurrence (e.g.: Pressing certain buttons, or typing into certain fields).

1. *PHP*

PHP is a server-side scripting language originally designed for web development, though now serves as a general-purpose programming language. PHP is The Force’s primary server-side programming language due to it being the language the team is most experienced with for interacting with databases. Through the implementation of PHP, The Force will be able to interact with and modify data stored on web servers in an efficient manner, allowing dynamic content to be accessed and displayed.

1. *Python*

Python is an object-oriented programming language normally used for developing web applications. Due to Python’s easy to learn syntax, high-level built in data structures, as well as flexibility and power thanks to its extensive library, it is an extremely valuable tool to have. Python will be used in accompaniment to the other languages discussed whenever it might provide an easier solution to a proposed issue.

* + 1. **Libraries / Frameworks:**

1. *Bootstrap*

Bootstrap is a HTML, CSS, and JavaScript framework used to aid in the development of a proper GUI. Bootstrap contains HTML and CSS based design templates for most necessary components of the GUI, and thus will be used to ease the development of The Force while also creating the visually appealing application possible for its users.

1. *jQuery*

jQuery is a JavaScript library which will be used to simplify many client-side scripting tasks such as document navigation, event handling, and AJAX implementation.

1. *Data Integration*

Data Integration is an Extract-Transform-Load (ETL) framework developed by Pentaho, which will be used to manipulate data into graphical representations. This will not be used for actual website implementation, but will instead be used by the server-side team to simplify the processing of data.

* + 1. **Relational Database Management Systems:**

1. *MySQL Workbench*

MySQL Workbench is a visual database design tool that implements SQL development, database creation, designs, and maintenance into a single IDE for MySQL database systems. The team will use it to create the database, as well as insert all necessary tables, before uploading this information to the server.

* + 1. **Web Server:**

The web server used for this application is Internet Information Services (IIS) 6.1, with PHP 7.0 and MySQl 5.6.26 implementation. Created by Microsoft, IIS is an extensible web server that supports HTTP, among other protocols, and can be used to host nearly anything on the Web. For the sake of flexibility and security, it will be used to host The Force.

* + 1. **Integrated Development Environments (IDE) / Editors:**

1. *CodeRunner*

*CodeRunner i*s an IDE and editor released exclusively for Mac, which supports and can run 20 languages of code. With regards to The Force, it will mainly be used for client-side coding (HTML, CSS, Javascript), though PHP and Python are both supported by the program.

1. *Sublime Text*

Sublime Text is a cross platform source code editor, which Python based plugin API should it be required. Due to its “Goto Anything” feature, Sublime Text allows for quick navigation of large files, making it easier for developers to find certain blocks of code. As it also natively supports several programming and markup languages, most notably those required for The Force, it allows for quick changes in larger quantities of code.

1. *Netbeans*

Netbeans is a software development platform written in Java. Though the Netbeans IDE is primarily intended for Java development, it also supports PHP and HTML5. As such, some members of the team will be using it for server-side development, due to previous experience with the platform.

1. *Eclipse*

Eclipse is an IDE that is, once again, written primarily in Java to allow cross-platform compatibility. Despite the fact that it’s mainly used for developing Java applications, plugins enable it to be used for the development of other types of applications as well. With regards to The Force, it will be used for develop PHP scripts, once again due to certain members’ experience with the platform.

* + 1. **Collaboration Software:**

1. *Slack*

Slack is a cloud-based team collaboration tool that allows individuals to join persistent channels (or chat rooms), as well as directly message individuals. Slack is the communication medium of choice for the team as a whole. Due to the simplicity of use, and the fact that people can easily be incorporated into multiple channels, Slack allows for direct, straightforward, and streamlined conversation between members. In addition, each channel includes a search feature, allowing important information to always be accessible without the need to save it somewhere else or scroll indefinitely till it is found.

1. *GitHub*

Github is web-based Git repository hosting service, and offers the same source code management as Git. Unlike Git, however, Github is not strictly a command-line tool, instead providing a web-based GUI, as well as bug tracking, task management, and other collaboration features. Github is used by the team to not only share and incorporate other members’ code segments, but is also used as the main documentation distributer for The Force. Documents can not only be easily uploaded and shared to the entire team, but documentation can also be posted and analyzed by other team members, allowing feedback before beginning proper implementation.

# 7 – Plan

# **7.1 – Activities**

### Activity 1 - Research

Research is described by gathering key information from various relevant sources. This gathered information contains only the necessary details that are pertinent to the project.

### Activity 2 - Planning

Before a project begins its development, a plan must be put in effect. This leads to an organized project where all members can track the work they have done and the work they have yet to complete.

### Activity 3 - Discussion

In any project that involves multiple members, discussion is key for an optimal development. Sharing ideas and reviewing each other's work can bring out effective ideas with a lower chance of errors.

### Activity 4 - Programming

Programming, both client-side and server-side, is key for the development of an application. The planning and ideas can be optimal but will only be valuable through effective programming.

### Activity 5 - Testing

It is impossible to truly know if a software is effective until it is properly tested. Many unforeseen errors can be uncovered through testing, and releasing an application before testing it can lead to a very negative experience for the user.

### Activity 6 - Debugging

The errors found through testing must be solved, which can be complicated when there is a multitude of code involved. Different segments of code must be analyzed for errors and programming solutions must be found.

### Activity 7 - Documentation

In order to be fully aware of the progress of a project, every action and plan of action must be well documented. It is only through organized documentation that a project can effectively be completed and ridden of errors.

# **7.2 – Artifacts**

### (1) Deliverable 0

Deliverable 0 is used not only to solidify system information and create a rough draft of the project’s Domain Model, including describing each major portion of said model, but it also serves to solidify each member’s role within the team. This artifact serves to outline what will be accomplished in the final project, as well as introducing the entire team.

### (2) Deliverable 1

Deliverable 1 serves to describe, in much greater detail than Deliverable 0, the project as well as its goals, requirements, domain model, constraints, resource allocation, and plan for implementation (among others). As this artifact covers most aspects of the application, including a schedule for the either project, it is incredibly important with regards to the development of said application.

### (3) Deliverable 2

Deliverable 2 serves to outline the architecture, class diagrams, and design diagrams in complete detail. The deliverable also incorporates reports on rapid prototypes, and analysis of the risks, scope, and estimates involved. Due to its technicality, this artifact is extremely relevant to the programmers of the team in order to create a proper idea of how the project should function.

### (4) Deliverable 3

Deliverable 3 serves as a summary of all testing cases previously reported, as well as what needs to be accomplished prior to the final stage of production. This artifact includes testing reports for all items and cases tested, and documentation for the system (including installation and user manuals), as well as a final cost estimate of the project.

### (5) Basic Project Structure Discussion

All members got together to discuss what features a proper course scheduler would include. In addition, methods of communication (i.e.: Slack), and which roles each team member would prefer were discussed. This artifact served as the beginning of the entire software development process, and created the initial foundation for the final project.

### (6) Initial Diagrams

Members of the team constructed various diagrams (including ER, Domain and class) in order to better visualize the relationships of the main sections of the project. The diagrams constructed in this artifact serve as the visual foundation for the programming development to come.

### (7) List of Features and Technologies Required

Once a solid foundation for general functionality was created, the team discussed exact features they believed the application should include. Once these features were determined, the technologies required to achieve this goals were discussed and decided upon. This artifact serves not only as a measure for what the team wishes to accomplish, but also as a way to determine which technologies will make accomplishing said goals as easy as possible.

### (8) Creation of Proper Programming Environment

When the technologies required have been decided upon, it is crucial to then set up said technologies to allow the team proper access to everything they need to complete the application. This artifact serves to ease the programming required in the long run, as establishing a proper programming environment (and thus proper project management) will make the task of completing the project more manageable.

### (9) Implementation Discussion

When all ideas and technologies have been determined and finalized, the team is then able to discuss plans with regards to proper implementation. This artifact involves team meetings and discussions, and serves to determine the practical ways in which programmers will now be able to begin proper development of the application, as well as implementation of the features previously mentioned.

### (10) Use Cases

Use cases describes all possible actions performed by a user, or an "actor" in a system. The work done in the backend by the system itself is not indicated through this type of model. This artifact allows the development team to see an overview of the user's experience and optimize it.

### (11) User Interface

A functional system can only be brought to its full potential through an optimal user interface. An effective UI is clear, concise, and without clutter. Before any programming begins, the user interface must first be designed in an effective manner. The client-side programming team must always keep this artifact in mind when implementing the layout of the application.

### (12) User Testing

It is important for the programming team to test their code, but it is equally crucial for the users to test. Having input from a non-technical perspective can inspire the development team to optimize the project for a better user experience.

### (13) Design of Script to obtain Concordia Class Data

Since a large amount of data is required for the integration of our project, this is an important artifact because it ensures that correct data will be populated in our SQL tables in the database.

### (14) ER Diagram creation

Once the data is obtained using the script and put into the database, the team created ER diagrams to show relationships between objects/tables derived from the domain model. This will help the team in the configuration of the manipulation of the data. This artifact is closely tied with the main functionalities of our system since the data used in the software must be well-represented and easily accessible.

### (15) Architecture Programming, Use Cases

The main part of the programming takes place within this artifact because the code is created which makes all the designed architecture and use cases functional. If this artifact did not exist, the system would not exist.

### (16) Scope of Project

Since the programming for this system is still ongoing, the scope of the project is subject to change. As the project progresses, the scope will change depending on what the team believes is still feasible to implement or not. This artifact will lead to the final product as element the team deems as nonessential/infeasible will be removed.

### (17) Test Cases

As functionality is added by programming, test cases are created to ensure that the newly added operations of the system are fully working. Most importantly, these test case are largely based on the use cases and also will test the functionality of the database as well. This artifact is essential for quality assurance and a working final product.

### (18) Performance

The Sub-team members responsible for QA will try out each test case on the system and determine whether or not the system operation is fully functional. This artifact will be the final confirmation that newly added system components are working and before they are put into the final polished product.

(19) Design of UI/Aesthetics of Application

The attractiveness of the product is important for the usability of the product. If users do not find the website esthetically pleasing, they will refrain from visiting it. On the contrary, if they like the aesthetics of the website, the users will appreciate the system more. This artifact ensure the final system is visually pleasing.

### (20) Debugging

During both technical and user testing, several errors can be uncovered but their source relies unknown. The programming team must discover the errors in the code and find effective solutions to prevent them from repeating.

# **7.3 – Project Estimates**

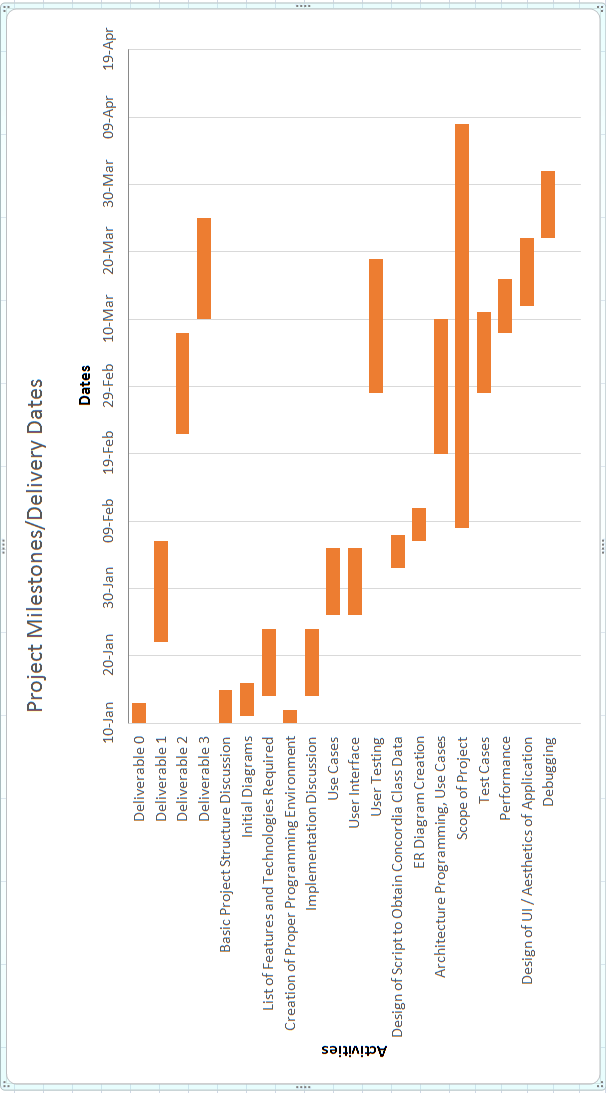
|  |  |  |  |
| --- | --- | --- | --- |
| **Task Name** | **Duration (Number of Days)** | **Total Cost (Hours)** | **Starting Date** |
| Deliverable 0 | 3 | 10 | 10-Jan |
| Deliverable 1 | 15 | 35 | 22-Jan |
| Deliverable 2 | 15 | 30 | 22-Feb |
| Deliverable 3 | 15 | 30 | 10-Mar |
| Basic Project Structure Discussion | 5 | 6 | 10-Jan |
| Initial Diagrams | 5 | 10 | 11-Jan |
| List of Features and Technologies Required | 10 | 10 | 14-Jan |
| Creation of Proper Programming Environment | 2 | 5 | 10-Jan |
| Implementation Discussion | 10 | 20 | 14-Jan |
| Use Cases | 10 | 15 | 26-Jan |
| User Interface | 10 | 15 | 26-Jan |
| User Testing | 20 | 65 | 28-Feb |
| Design of Script to Obtain Concordia Class Data | 5 | 6 | 02-Feb |
| ER Diagram Creation | 5 | 6 | 06-Feb |
| Architecture Programming, Use Cases | 20 | 90 | 19-Feb |
| Scope of Project | 60 | 20 | 08-Feb |
| Test Cases | 12 | 15 | 28-Feb |
| Performance | 8 | 20 | 08-Mar |
| Design of UI / Aesthetics of Application | 10 | 20 | 12-Mar |
| Debugging | 10 | 20 | 22-Mar |
| **Total** | 90 | 440 | N/A |

**\*\*** Estimates were made based on time required to create, as well as based on each team members’ individual work ethic. Based on the activities assigned to each member (and the artifacts said activities produce), an appropriate amount of time was estimated, taking into consideration the number of people striving to achieve said artifact, and the time they have available for said tasks. For all artifacts not yet created/achieved, re-estimation might occur if it is discovered that the time provided is inefficient for the members responsible to complete their task. In addition, the opposite can be said; re-estimation might occur if the task is easier than expected, thus saving time in that regard and allowing a reallocation of time to another issue. Re-estimation would also be required in the case that additional components not yet listed were designed and implemented. In such as a case, time would need to be allocated to these new tasks and the time provided to our sections might decrease as a result.

# **7.4 Activities Assignments**

|  |  |  |
| --- | --- | --- |
| Member Name | Main Role within Project | Activities Assigned |
| George Theophanous | Team Leader  Leader of Quality Assurance | Discussion, Planning, Research, Testing |
| Stefano Pace | Leader of documentation  Assisting in both front and server-side programming | Discussion, Planning, Research, Documentation |
| Julian Ippolito | Leader of client-side programming | Discussion, Research, Programming, Debugging |
| Joey Tedeschi | Documentation  Assisting in both front and server-side programming | Discussion, Research, Documentation |
| Adam Arcaro | Documentation  Assisting in both front and server-side programming | Discussion, Research, Documentation |
| Georges Mathieu | Leader of server-side programming | Discussion, Research, Programming, Debugging |
| Olivier Cameron-Chevrier | Server-side Programmer | Discussion, Research, Programming, Debugging |
| Marc-Andre Dragon | Server-side Programmer | Discussion, Research, Programming, Debugging |
| Jordan Stern | Front-end Programmer | Discussion, Research, Programming, Debugging |
| Hasan Ahmed | Front-end Programmer | Discussion, Research, Programming, Debugging |
| Kuan Jiang | Quality Assurance | Discussion, Planning, Research, Testing |

# **7.5 Schedule**



# **7.6 Risk**

1. *Lack of Communication and Time Allocation*

Seeing as most members of the team have never worked together, some efficiency is lost while trying to establish a proper team dynamic, and effective communication between teammates. All the members of The Force also have requirements outside of the team, including other classes, work, or extracurricular activities. As such, time management is extremely important and it’s possible that some features will need to be left out due to lack of time.

1. *Relearning required languages*

Though most members of The Force have experience, whether basic or advanced, with web development, not everyone has recently been practicing the various languages being used. As such some time was lost at the start of the project in order to allow members to familiarize themselves with the languages being used, as unfamiliarity would lead to lost time in the future when more complicated algorithms and features are required.

1. *Server Side Security*

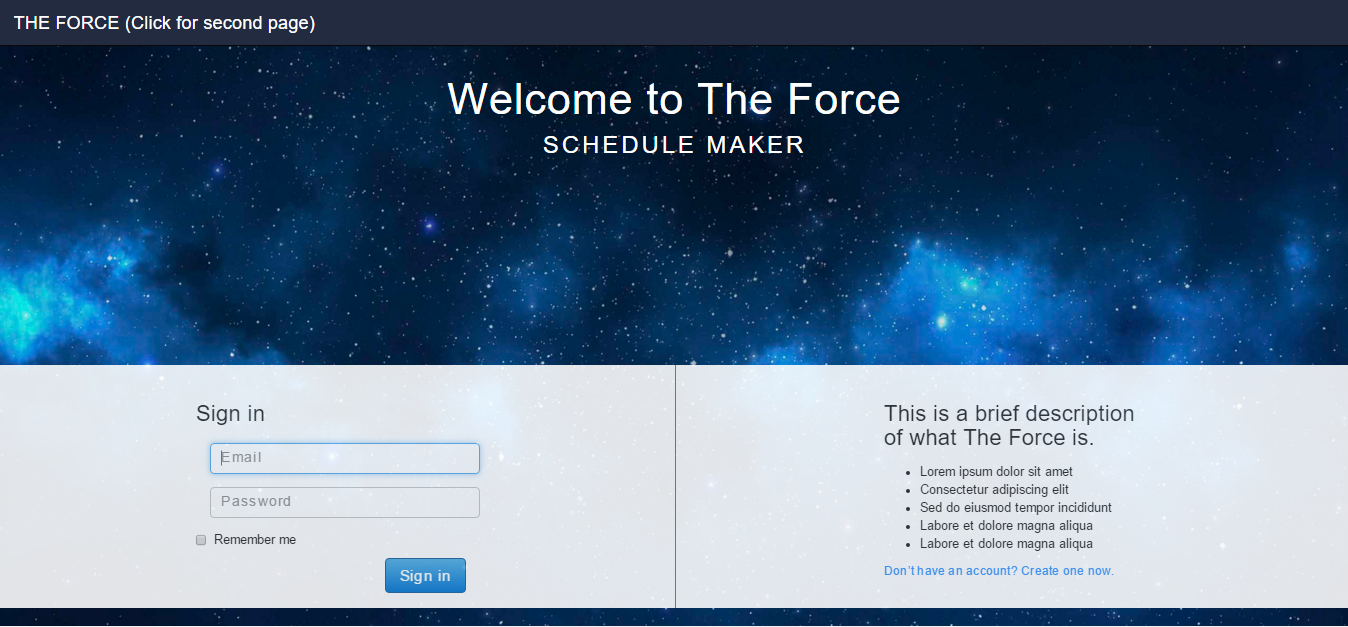
As the server is being hosted online, it is imperative that proper security protocols be applied. Without proper testing of the server and its limitations, it could eventually be revealed to the team that the server might not be able to store all the data required, or certain requests might not be handled properly. Should errors begin occurring with regards to the server and data management, the entire project would fall apart.

1. *Miscommunication between client and server side*

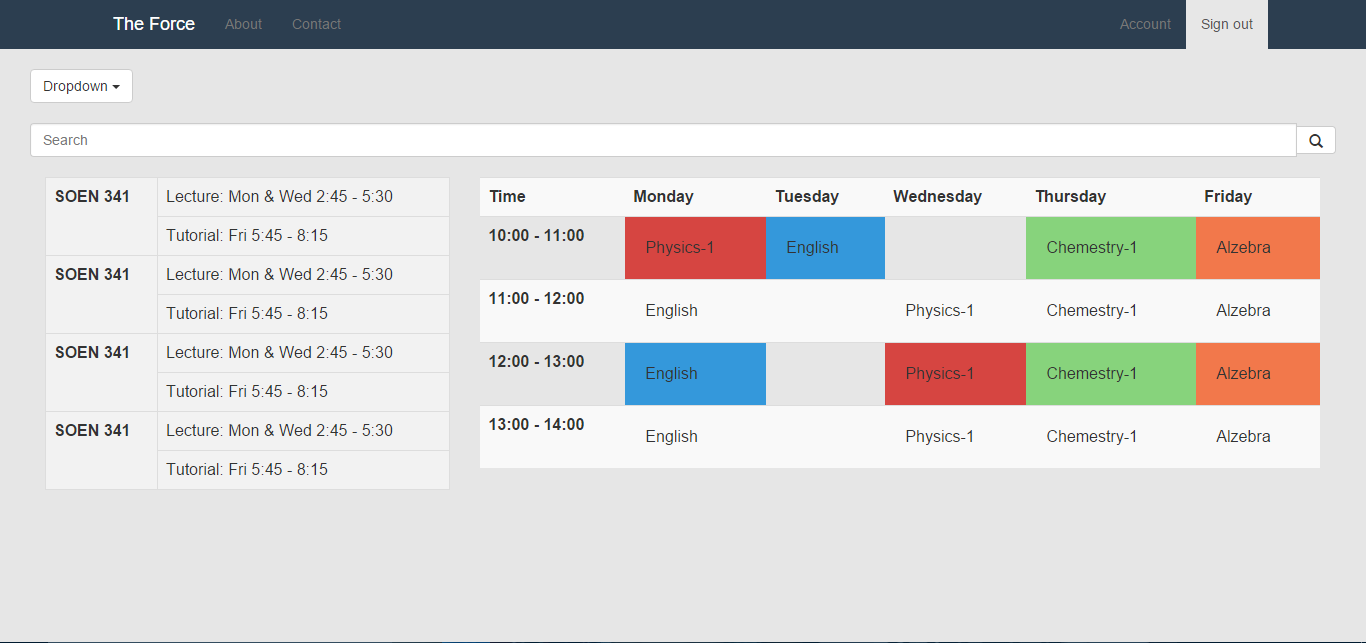
Should inputs provided by the user not be properly registered, the entire project would cease to function properly. For this reason, risks associated with the GUI (such as buttons not working, or information not being passed to / retrieved from the database) need to be treated with the utmost urgency.

# 8 – Prototyping

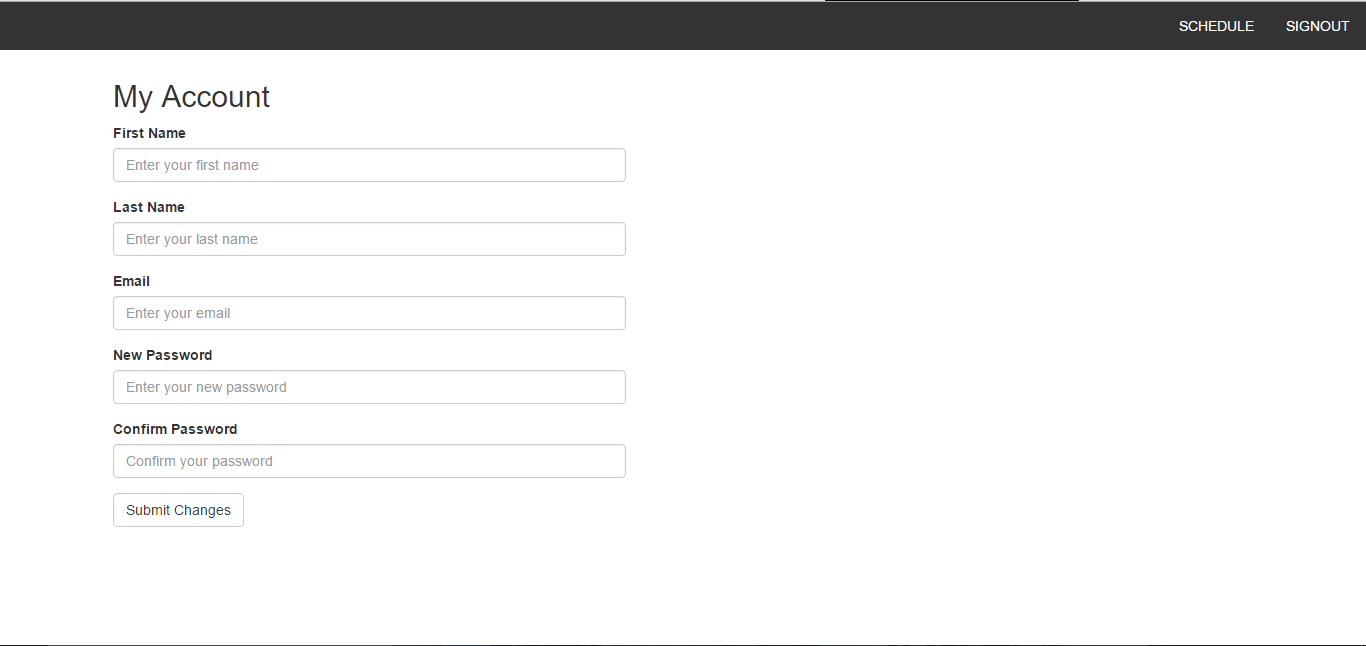
The following will describe work undertaken in development of our prototype “The Force”.



The screenshot above shows the home page of the website. Here, you will be able to sign in with the email associated to your Concordia Student account and your password. There is also a feature to “remember me” this way you can sign in faster every time you open the website. If you do not have an account, this means that Concordia has not created one for you yet. This can be either due to the fact that you are a new Student. If that is the case, you will be able to create an account and Concordia will populate all the necessary information to your profile (i.e: Student Transcript, courses taken etc.)



The screenshot above shows the main page of the website. Here, you can search for courses and add them to your schedule. By clicking on courses already added, you can also remove them from your schedule or browse for other sections available at different times. The search bar on top allows a student to manually search for the course they are look for by entering the course code. An autocomplete feature will kick in once they start to type, allowing for fast filtering and searching. The Header will include links to the “About Us” & “Contact Us” section of the site, which have not been developed yet. As well, there is a link to the “Account” section, which will be described below and an option to sign out.



The screenshot above shows the “My Account” portion of the website. Here, any student can view and edit any of the information associated to their account. As of right now, the team has incorporated First Name, Last Name, Email address, and the options to change password. If the password do not match, the user will be informed through a message on the screen. In the future, the team will add more of the student information here such as Student ID, Transcript, courses taken, etc. At the top, the Header has yet to include the section previously mentioned, but it includes a link “Schedule” which navigates the user back to the main page shown in diagram 2.

On top of this, the Server Side Team has created scripts in Python to extract all pertinent data from the data from Concordia’s website related to the course sequence for Software Engineers. They have all created all of the SQL tables for the values specified in each class of the domain model, and incorporated them into the host server. Last they created the following ER diagram to aid in their coordination between SQL tables and PHP scripts.

The website can be accessed at the following link for future inspection: <http://wolfcall.ddns.net:8085/>

