Concordia University

**Department of Computer Science**

**and Software Engineering**

**Software Process**

**SOEN 341/4 S --- 2016**

**Project Design Document**

**Team: The Force**

|  |  |
| --- | --- |
| **Team information** | |
| **Name** | **SID** |
| **Stefano Pace** | **27454716** |
| **Julian Ippolito** | **27419112** |
| **Adam Arcaro** | **27459157** |
| **Joey Tedeschi** | **27513062** |
| **Georges Mathieu** | **26863477** |
| **Hasan Ahmed** | **27546467** |
| **Marc-Andre Dragon** | **27721579** |
| **Jordan Stern** | **27400144** |
| **George Theophanous** | **27400578** |
| **Olivier Cameron-Chevrier** | **27228805** |

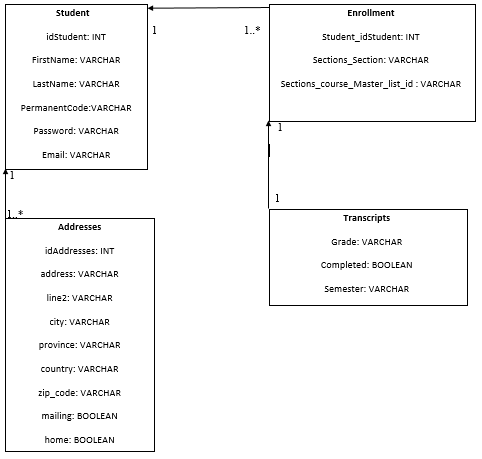
# 4. Detailed Design

This detailed design will include the logical back-end organization of “The Force” thus far.

Subsystems

User management subsystem

1. **Detailed design diagram**



This subsystem includes all student information. It allows for the creation of an account on “The Force”. The student may first sign up, and then subsequently sign in with their credentials. Furthermore, the subsystem stores relevant information regarding their transcript and their student record. This allows the subsystem to communicate clearly with the course registration subsystem. The Student class will be references by “Enrollment” in the database, which in turn will be referenced by the “transcripts” class. An instance of the “transcript” class refers to a course the student has taken. Any superfluous information about the student that is not directly needed to enroll in a course is stored in a separate class called “Addresses” which references the “Student” class. The address table is included in the design for two reasons. The first reason is so the user can maintain his/her current address through this portal (stretch goal of the system). The second was to account for this system’s storage to draw information for other systems the school could use. This separation allows for easier manipulation of a user. The “enrollment” class is what will allow the user management subsystem to communicate with the course registration subsystem.

2. **Unit description**

**Class Student**

* idStudent (INT): stores the student id number
* FirstName (VARCHAR, String): stores the student’s first name
* LastName (VARCHAR, String): stores the student’s last name
* PermanentCode (VARCHAR, String) : stores the student’s permanent code
* Password (VARCHAR, String) : stores the student’s password for authentification purposes.

All above information is what is vital to the student’s ability to registering for courses and singing in.

**Class Addresses**

* idAddresses (INT): Because a student may possess many addresses, idAddresses stores an id number which may help refer to the same student without confusing the address instances.
* address (VARCHAR, String) : stores the student’s address
* line 2 (VARCHAR, String) : stores the student’s alternate address
* city (VARCHAR, String) : stores the student’s city
* province (VARCHAR, String) : stores the student’s province
* country (VARCHAR, String) : stores the student’s country
* zip\_code (VARCHAR, String) : stores the student’s zip code (postal code)
* mailing (BOOLEAN) : determines whether or not the given address is the mailing address to send any important documentation.
* home (BOOLEAN) : stores whether or not the address is a home address or not

**Class Enrollment**

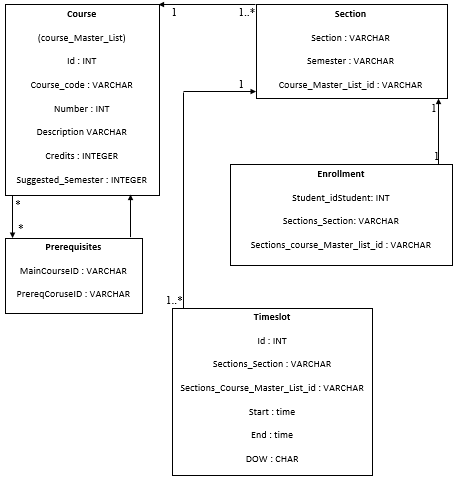
* Student\_idStudent (INT) : stores a student’s id number. This allows the instance of “enrollment” to associate with a specific student.
* Sections\_section (VARCHAR, String) : stores the section the student is enrolled in.
* Sections\_course\_Master\_list\_id (VARCHAR, String) : stores the id number of the course the student is enrolled in from the database. This allows for the program to determine which courses have been completed or not.

**Class Transcripts**

* Grade (VARCHAR, String): stores the student’s grade in the given course
* Completed (BOOLEAN) : stores whether or not the student successfully completed the course or not.
* Semester (VARCAHR, String): stores the semester in which the student took the course.

Course subsystem

1. **Detailed Design Diagram**



The course subsystem contains all information regarding course numbers, IDs, and times. Having an entire subsystem where this information can be stored allows for a clean separation of tasks between the two subsystems. Although a student will enroll to a course from their point of view, the system sees it as an enrollment in a section. This is because different sections have different timeslot instances. This allows the system to simultaneously enroll the student while checking for time conflicts. It is important to note that any information regarding the course name and number starts at the course class. From there, you can even find the suggested semester. As you go further away from the course class, there is a “breadcrumb trail” that can always lead you back to it easily. Class Section contains an ID that matches to a class in the master list. Enrollment and timeslot classes have the same course name and number, although we append “Section\_” at the front, to force the path through the section class, without accidentally accessing every single section in the course. On top of this information, they contain the section ID in order to not pass through and get enrolled into the wrong section.

1. **Unit Descriptions**

**Class Enrollment**

* See “User management subsystem”
* This class serves as a “UI” between the two subsystems
* Sections\_Course\_Master\_List\_id (VARCHAR, String) : Refers to the id of a course. This id is first matched to a section along with Sections\_Section (VARCHAR, String) to match to a section the student is enrolled in.

**Class Section**

* Section (VARCHAR, String) : Stores the section number of a course the student is enrolled in.
* Semester (VARCHAR, String) : Stores the semester of the course the student is enrolled in.
* Course\_Master\_list\_id (VARCHAR, String) : Stores the ID of the course the student is enrolled in. This is matched to a specific ID in the database (Course class).

**Class Course**

* Takes the form of Course\_Master\_list in the database. Contains all information regarding courses.
* Id (INT) : Stores the ID number of a course.
* Course\_Code (VARCHAR, String) : Stores the course code in a string (4 letters)
* Number (INT) : Stores the course number (3 digits).
* Description (VARCHAR, String) : Stores the description of a course so that the student may know what they are chosing.
* Credits (INTEGER) : Stores the number of credits the course provides.
* Suggested\_Semester (INTEGER) : Stores the suggested semester of the year for the student (between 1 and 3). This allows to provide the student with a recommended course sequence.

**Class Prerequisites**

* MainCourseID (VARCHAR, String) : Stores the course name and number of the course the student is trying to enroll in.
* PrereqCourseID (VARCHAR, String): Stores the course name(s) and number(s) of the course(s) the student should have previously completed in order to enroll to “MainCourse”.

**Class Timeslot**

* ID (INT) : contains the student number of the student taking a course in that given timeslot.
* Sections\_Section (VARCHAR, String) : Stores the section number of the course of the timeslot the student is enrolled in.
* Sections\_Course\_Master\_List\_id (VARCHAR, String) : Stores the course name and number of the section of the timeslot the student is enrolled in.
* Start (Time) : the starting time of the timeslot.
* End (Time) : The ending time of the timeslot.
* DOW (CHAR): Day of the Week represented by a single character.

# 6. Estimation

## Revised Project Estimates

The following is a revised cost estimate for the project. Upon review of what has been completed by the Project team so far, as well as challenges/risks encountered, the cost estimates have changed. In some cases, they have risen and in others they have dropped.

\*Numbers in Blue signify costs that have dropped

\*\*Numbers in Red signify costs that have risen

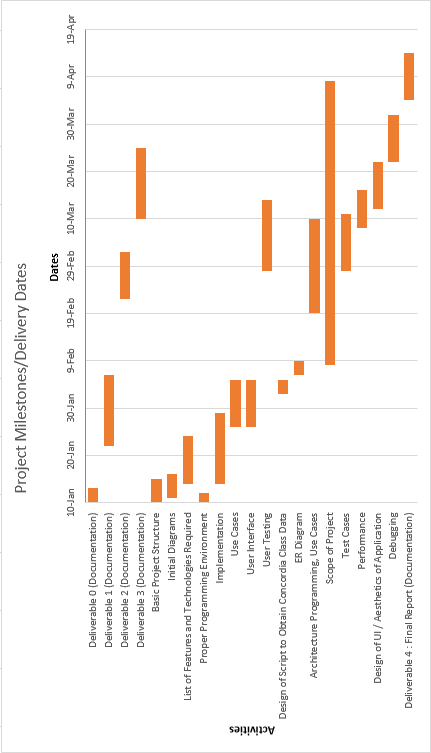
\*\*\*Highlighted Dates signify changes in start Dates

|  |  |  |  |
| --- | --- | --- | --- |
| **Task Name** | **Duration (Number of Days)** | **Total Cost (Hours)** | **Starting Date** |
| Deliverable 0 (Documentation) | 3 | 10 | 10-Jan |
| Deliverable 1 (Documentation) | 15 | 35 | 22-Jan |
| Deliverable 2 (Documentation) | 10 | 25 | 22-Feb |
| Deliverable 3 (Documentation) | 15 | 30 | 10-Mar |
| Basic Project Structure | 5 | 6 | 10-Jan |
| Initial Diagrams | 5 | 10 | 11-Jan |
| List of Features and Technologies Required | 10 | 10 | 14-Jan |
| Proper Programming Environment | 2 | 5 | 10-Jan |
| Implementation | 15 | 35 | 14-Jan |
| Use Cases | 10 | 15 | 26-Jan |
| User Interface | 10 | 15 | 26-Jan |
| User Testing | 15 | 55 | 28-Feb |
| Design of Script to Obtain Concordia Class Data | 3 | 6 | 02-Feb |
| ER Diagram | 3 | 6 | 06-Feb |
| Architecture Programming, Use Cases | 20 | 75 | 19-Feb |
| Scope of Project | 60 | 15 | 08-Feb |
| Test Cases | 12 | 15 | 28-Feb |
| Performance | 8 | 20 | 08-Mar |
| Design of UI / Aesthetics of Application | 10 | 15 | 12-Mar |
| Debugging | 10 | 30 | 22-Mar |
| Deliverable 4 : Final Report (Documentation) | 10 | 30 | 4-Apr |
| **Total** | 95 | 463 | 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.

## Revised Schedule & Gantt Chart

The initial Overall Man-hours for the project was 440. The revised Overall Man-hours for the project is now 463. This is an increase of roughly 5%. As a result, the team has updated the schedule to the following:



# Rapid Prototyping and Risk

## Prototype

This interface mock-up provides insight into the first design concept for the project. It shows the

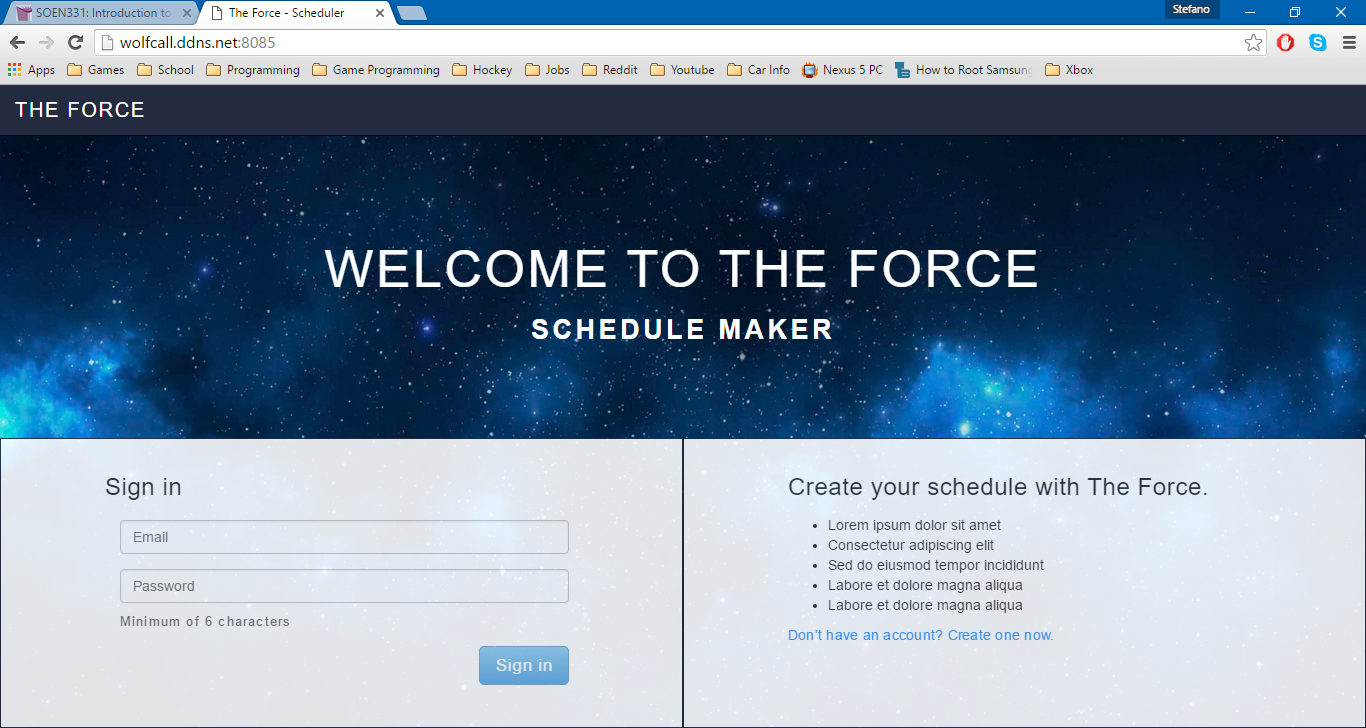
Register module which provides information about the courses as well as allowing the user for register them.

Other modules would show the current schedule for registered courses as well as the user’s profile information.

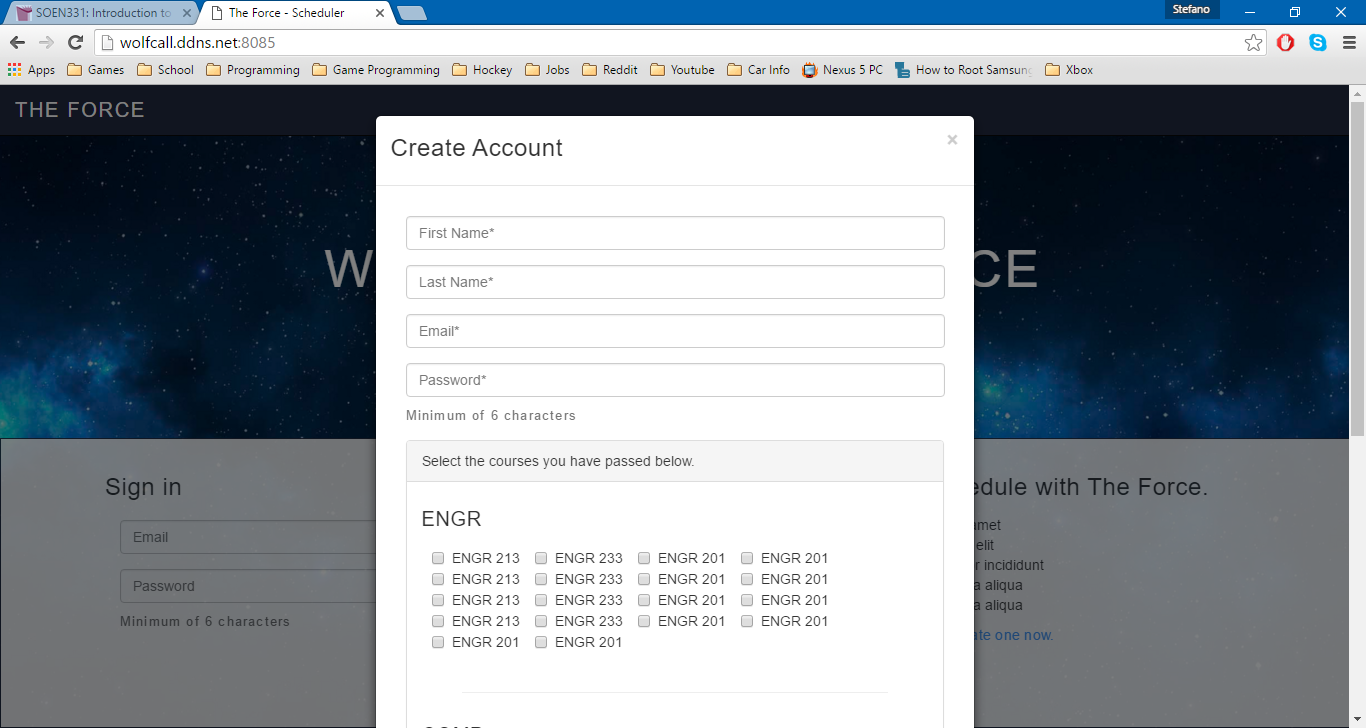
In order to accelerate the development of the project, continuous prototypes will be produced.

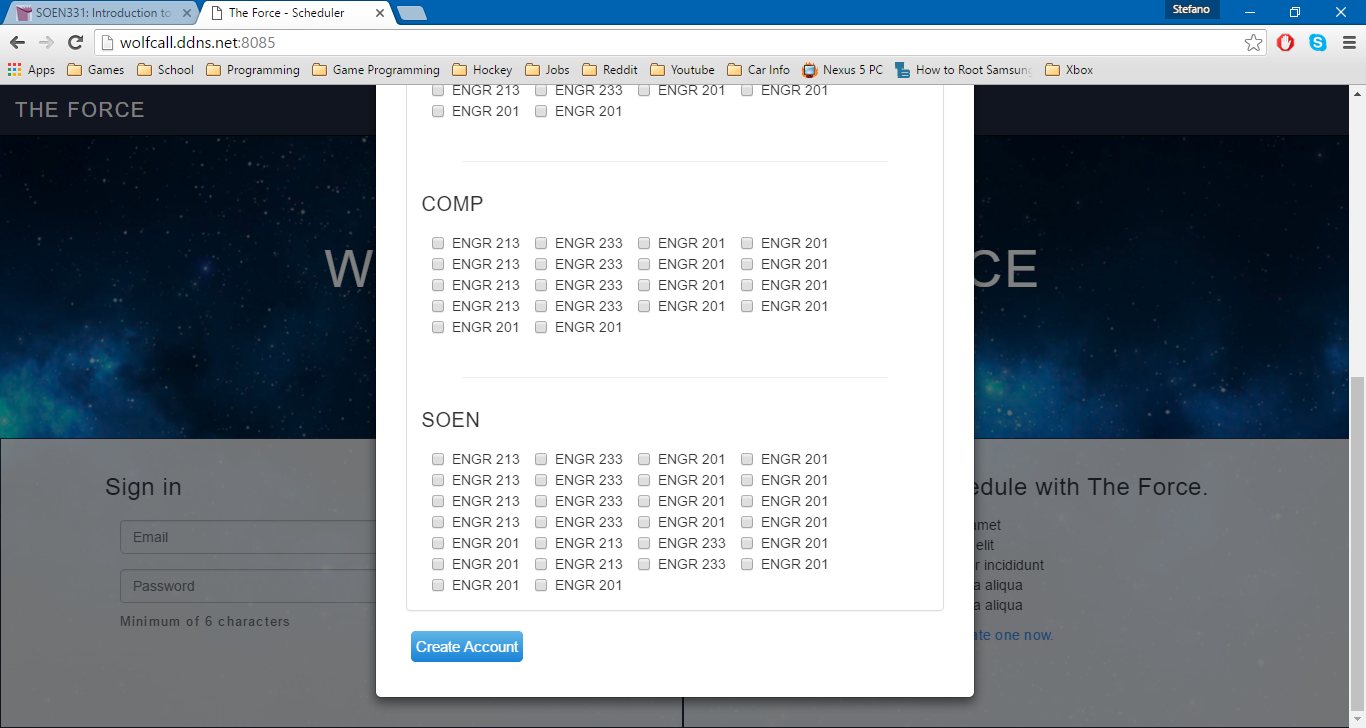
The rapid prototyping method will allow the programmers and designers to adapt and implement new ideas faster, as well as to accelerate the learning curve of the programming language.

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

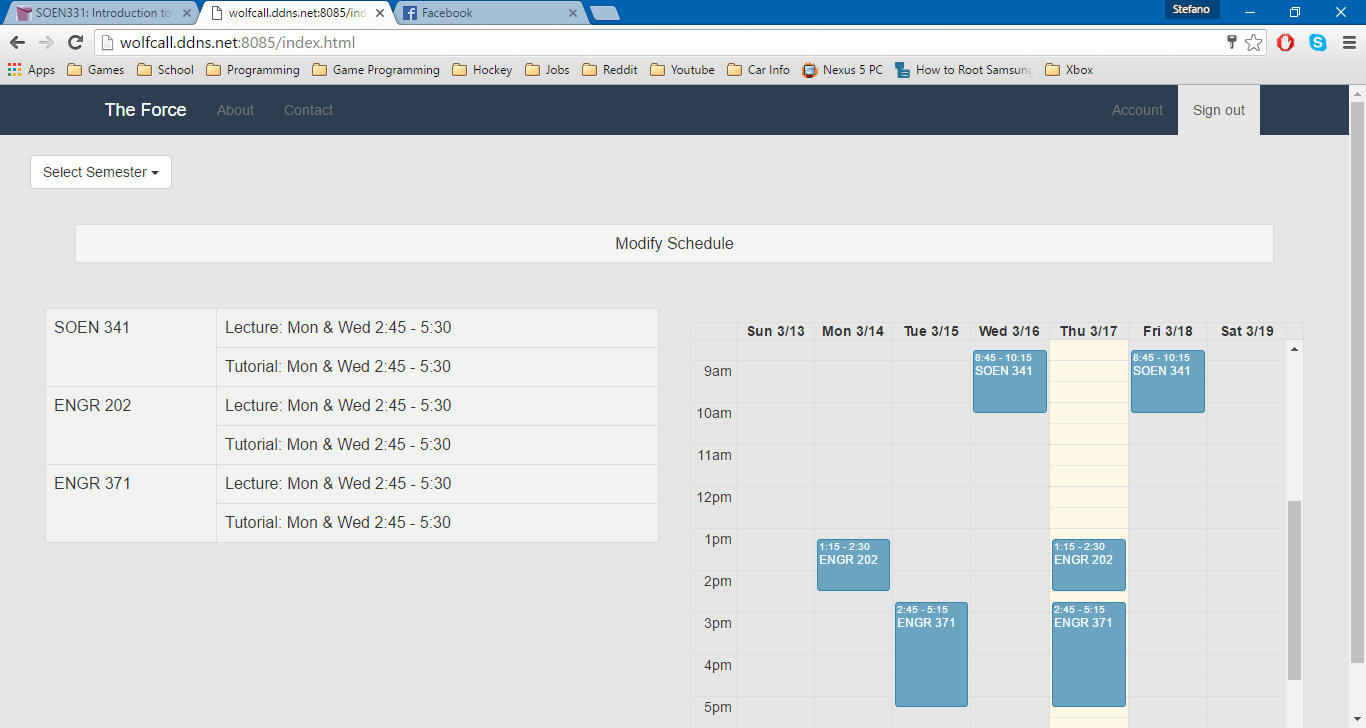


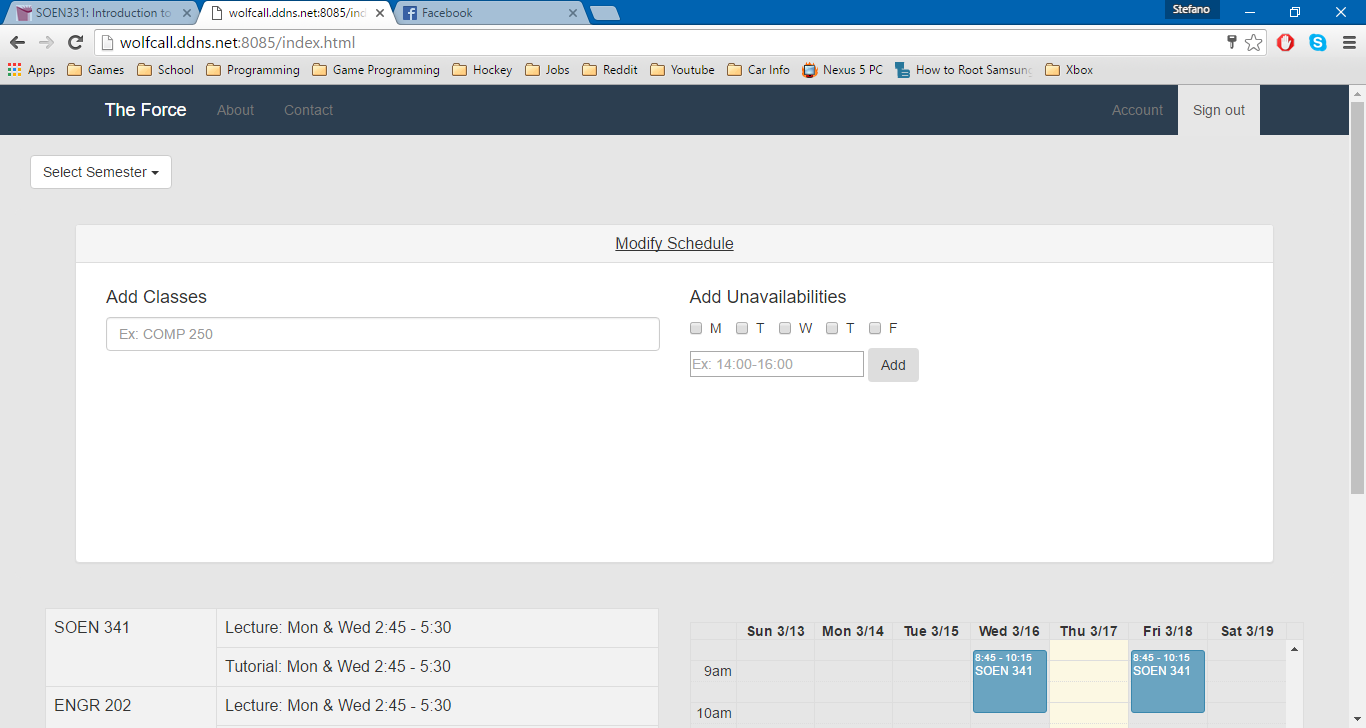
The screenshot above shows the updated home page of the website. Here, you will be able to sign in with the email you used when you created the account. If the information you entered does not match anything in the database, then you will not be able to sign in. It is suggested that you use the email that is associated to your Concordia Student account. If you do not have an account, you can create a new account.





If you select the “create an account” option, a dynamic window will appear. It will prompt the user to enter all pertinent information such as Name, Email, Password and Courses taken. The user will also have to check all the courses they have taken and passed, so they can be registered as pre-requisites. Once all information is entered, it will be saved to the database and you will be redirected to the main page of the website.



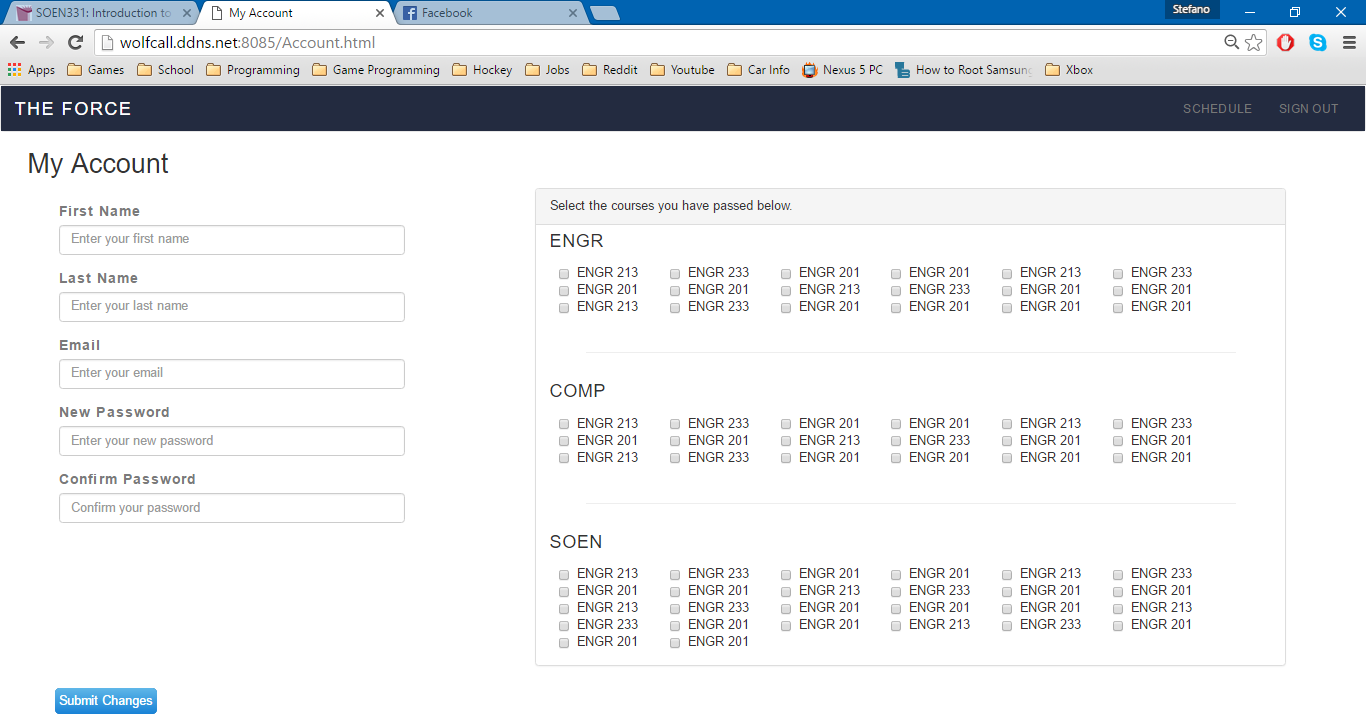


The screenshot above shows the main page of the website. Here, you can search for courses by entering the course code and add them to your schedule. You can specify the time and date constraints to your search so it will provide you with filtered search results. An autocomplete feature will kick in once they start to type, allowing for fast filtering and searching.

The courses added by the user are added to database and are associated using am ID. Every user on the website is given a unique ID, so there is no way to display the wrong data.

By clicking on courses already added, you can also remove them from your schedule or browse for other sections available at different times. If the user tries to add a course that a student already has in their schedule, they will be prompted if they want to overwrite the course. If they select yes, the data will overwrite in the database. If they select no, then the data will stay the same.

The Header 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 (mentioned above). If the password do not match, the user will be informed through a message on the screen. When the user changes any information here, it will be overwritten in the database.

## Risk

Four risks were initially identified within the scope of this project.

The first was the Lack of Communication and Time Allocation. The concern was that since most of the member of the team have never worked together before, the overall efficiency would be slow at the beginning of the project while a proper team dynamic was formed. This proved to be a non-issue, as the team gelled quickly and it allowed the team to complete the early tasks in shorter times than expected. As a result, the team decided to allocate more time to tasks deemed more challenging than initially expected (i.e. Implementation, Relearning required Languages). Time management remains extremely important to our team and it’s possible that some features will need to be left out due to lack of time.

The second risk was relearning required programming languages. The concern was that members of The Force, whether they had basic or advanced knowledge of web programming, were not practicing recently so they had to take some time to re-familiarize themselves. As mentioned above, since the early tasks were completed quickly, more time was allocated for this.

The third risk was the security of the server. The concern was that proper security protocols had to be applied. Without proper testing of the server and its limitations, there was a fear that it could eventually be revealed that the server could not store all the data required, or that certain request might not be handled correctly. This would lead to the collapse of data management and as a result, of the entire project. With the work done so far, we have taken extra caution in ensuring that this does not happen, so there is no change with regard to this risk on the scope of the project.

The fourth and final risk was the miscommunication between the client and server side. The concern was that if the user would not properly register, then the entire project would stop functioning properly. Therefore all risks associated to the GUI’s (button’s not working, or information not being passed/retrieved from the database) need to be top priority. With the work done so far, the team has not encountered any problems with the communication between the client and server side, so there is no change with regard to this risk on the scope of the project.