Concordia University

Department of Computer Science

and Software Engineering

Software Process

SOEN 341/4 S --- 2016

Project Testing and Delivery 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 |

Introduction

Before delivering an application to a customer, one should make sure that a thorough testing report has been completed. This document provides explanation of all tested items as well as untested items of interest. These items of interest will require more testing in the future. Details of all the test cases current finished are also included. They are unit testing, requirements testing, stress testing and last but most importantly security testing.

When delivering an application to a customer, clear instructions must also be provided. The instructions provided in this document should cover both installation and use of the system. The Installation Manual describes the steps to set up necessary hardware and software for the system before use. Next, the User Manual details the features of the system and provides instructions on how they work. Both of these manuals contain screenshots of the steps to help visualize and facilitate the process for the user and administrator.

The third part of this document includes the final cost estimate of the system. It breaks down further changes to the project Estimates table as well as the reasoning behind all changes. Another updated Gantt chart is provided, and pie-charts are used to show a change in the distribution of the number of hours per activity. Last but not least, an extra section is included, showing further changes to the use case tables after continued implementation of core functionality.

1. Testing Report

1.1 Test Coverage

1.1.1 Tested Items

**Tested Units**

With regards to item testing, all requirements were grouped into appropriate units/classes based on similarity and interdependence of function (E.g.: Login and Logout). Each requirement is, in actuality, a use case previously outlined in Deliverable 1 and improved upon in Deliverable 2. In addition, a unit was created consisting of a new feature/use case: “Create Student Account”. This unit consists of the use case “Create Student Account”, which was not included in previous deliverables due to its relative newness. More information on this use case can be seen in section 4.

All requirements below are considered to be part of the core functionality of The Force, and thus are given a higher priority in comparison to something such as admin access and features (refer to 1.1.2 for further information). All units, and the core features/requirements they represent, can be seen in the table below. The test cases associated with each unit and requirement is numbered, and can be seen in further detail under section 1.2.2.

|  |  |  |
| --- | --- | --- |
| Unit/Classes | Related Requirements | Test Case(s) |
| Login | UC01 | 1.1 – 1.2 |
| Logout | UC02 | 2.1 – 2.2 |
| Create Student Account | UC16 | 16.1 – 16.8 |
| Edit Student Information | UC03 | 3.1 – 3.3 |
| Edit Courses | UC05, UC06, UC07 | 5.1 – 5.8, 6.1 – 6.4, 7.1 – 7.3 |
| Schedule Generation | UC08, UC11 | 8.1 – 8.3, 11.1 |

Login

The login procedure is incredibly important with regards to The Force as a whole. In order to access the scheduler, student information, or administrative access, a proper email and password combination is required. The login will only be successful if the combination of this email and password previously existed in the database (associated to a single user). Should this combination not exist, access to any further functions would be denied and the email and password fields would be cleared. Login is a unit that would be used several times, incredibly often, by every user of The Force. As such it requires extensive testing, as it contains the requirement that allows all other features to be accessed effectively.

Logout

Logging out is needed in order to allow a student or administrator to exit their account once they’ve completed their actions. Logging out ensures safety and security for the user’s account should they be sharing the device they used The Force on with others. Without logging in and logging out, all functionalities provided by The Force become irrelevant, thus it is a very high priority. In addition, all users will heavily use the logout feature, just as they use the login feature. As such, the logout unit must be tested in order to ensure the safety of all users, otherwise it runs the risk of massive security breaches across multiple users.

Create Student Account

In the case that a student wishes to create a schedule, it is required they have an account in place. For this reason, the creation of student accounts is another incredibly important requirement for The Force. Anytime a student without an account already in place would like to use The Force, they would need to make an account. This unit will be tested by following the instructions provided upon requesting to create an account, and then confirming whether or not a new entry was created in the databases with all the appropriate information. In the case that an account is made properly, the student will be redirected to the main page with their current schedule and courses.

Edit Student Information

Based on the student currently using The Force, the appropriate account information will be displayed on the Accounts page. From this page, the student will be given the option to change their password should they like. This will be tested through a mock-account, and the new password can be checked through accessing the database after the change, or simply attempting to re-enter the application after logout using the new password. This feature is more so a novelty for the student/administrator using it, but it is important due to the comfort it can provide a user as well as the added security provided thanks to the option to change one’s password.

Edit Courses

Course addition is an incredibly vital part of the entire schedule generation process. Without the option to manually add courses, students would be left without any true control over the courses they take. Every student using The Force needs to add courses, and thus it is considered extremely important with regards to the application. Courses added would need to ensure no conflicting time slots, as well as ensuring that the course is not allowed should the necessary co- and prerequisites not be present. In addition to choosing a course, students are expected to choose an appropriate section as well, which will also be tested for time slot conflicts before being added to their schedule. Should these two options misbehave, students will be unable to add courses to their schedule, nor edit the sections they are in, thus invalidating the entire schedule generation process, and the application as a result.

Deleting a course is of an equal importance. Should a student wish to remove a course they had entered in their schedule during the registration period (possibly due to a new, more important course conflicting with the time), or during the semester for some other reason, it is imperative that the option function perfectly. This option will once again be used by all students, and can be tested by simply removing a course from a schedule and ensuring that the necessary updates take place

Schedule Generation

Schedule generation is, in fact, the primary purpose of The Force, and as such may very well be the most important unit present (aside from login/logout). Once courses have been added or removed according to the student’s wishes, their schedule will be generated and saved. This can be tested by simply observing whether the schedule is still present for a given semester upon refresh of the main page. Seeing as it is the entire purpose of the application, extra precautions must be taken.

Assuming no courses were manually added/removed, or if a student doesn’t care for the courses they have already added, a schedule can also be auto generated based on the courses said student needs to take next in their sequence. Though this may not result in a student’s ideal schedule, the option to edit sections and remove courses will still be present after the generated schedule has been accepted. Auto generation of a schedule is far more complicated than manual generation, as a student will normally know what courses they need to take next and can physically see and avoid time conflicts. As this is not the case for auto generation, extra precautions are placed and need to be tested (e.g.: Overlapping timeslots, registering for previously completed courses, registering for courses without the pre- or co-requisites required). This can be tested through observation of the generated schedules, and addressing any issues that may arise. Both methods of schedule generation, should they not be implemented properly, could jeopardize the entire purpose of the application, and thus are of the utmost importance.

Once a schedule has been generated, a student may give his or her approval and save it to the semester they are currently creating it for. Once saved, the schedule, as well as the courses they are currently enrolled in, will be visible upon login and on the main student page. All students will have access to this saved schedule, and without properly implementing the save feature, a student would never be able to store their schedule for later use (effectively removing the need for one). Though the program would function without this feature, it would serve no real purpose, as students need to be able to see the classes and sections they are currently, and will be, enrolled in.

**Tested Requirements**

The requirements displayed above were tested for one of 2 reasons:

1. They are considered integral to the development of web application project as a whole, and thus were tested in order to gain an understanding of the progress being made as a whole (e.g.: Login, add course).
2. They were lesser features, which were ready for testing (e.g.: View / Edit Profile).

Any explanation for the necessity of testing of these requirements can be found below, as well as an indication of their level of importance in comparison to the other requirements being tested

* Login (Use Case 1)
  + This requirement is considered to be of a **high importance** level, as should the login feature fail, further access to the application would be restricted. Not only would further testing be impossible, but any use of the program would be as well.
* Logout (Use Case 2)
  + This requirement is considered to be of a **high importance** level, as without the option to logout, other unique users would not be able to utilize the application from a shared device. In addition, overall security for each user would diminish due to the fact that their accounts would, in a sense, always be available to anyone who managed to get the device said individual used.
* View / Edit Information (Use Case 3)
  + This requirement is considered to be of a **high** **importance** level, as the ability to view ones account information can be important (e.g.: Seeing which courses have already been taken), as well as editing something such as a password in the case of security risks. In addition, it is important to be able to view any errors in a student’s personal or academic information, thus allowing the proper help to be notified.
* Add Course (Use Case 5)
  + This requirement is considered to be of a **high importance**, as it serves as the backbone for manual schedule generation. Without the ability to manually add a course to ones schedule, students would be forced into only auto-generating their schedule without the ability to then add one which they would prefer (as manual generation simply would not exist).
* Change Course Section (Use Case 6)
  + This requirement is considered to be of a **high importance** level, as without it a student would be restricted to the section either provided to them by the auto-generator, or which they themselves chose while initially creating their schedule. Should a situation arise where a student wanted to switch their section, perhaps into a better-timed one, or in order to avoid a conflict with another course, this feature is required.
* Remove Course (Use Case 7)
  + This requirement is considered to be of a **high importance** level, as without it students would be unable to remove course automatically assigned to them, or which they lost interest in. This feature is required when editing one’s schedule, as without it new courses could never be added (assuming the student has reached a “maximum” number of courses, or they’ve decided to no longer take a particular course).
* Generate Schedule (Use Case 8)
  + This requirement is considered to be of a **high importance** level, as it represents the entirety of the auto-generated schedule feature. Without this feature, students would be restricted to only creating schedules manually, even though they may be indifferent to the outcome of their schedule; they may simply wish to follow their sequence (and perhaps make changes after auto-generation takes place).
* View Saved Schedule (Use Case 11)
  + This requirement is considered to be of a **medium importance** level, as it doesn’t represent a core functionality of the application, nor does it provide additional user security. Even so, the entire purpose of creating a schedule, aside from enrolling in classes, is to be given the option to refer to said schedule at a later date (perhaps to check course times, or locations). Thus though it is not a main pillar of The Force, it is relevant nonetheless.
* Request New Account (Use Case 16)
  + This requirement is considered to be of a **high importance** level, as every student is required to make an account before access to The Force’s features are provided. This is equally as important as the ability to login, as it serves as the precursor to it. Without the ability to create an account, a student would never be able to eventually login and begin creating their schedule.

1.1.2 Untested Items of Interest

Two main units were not tested, but remain relevant nonetheless: Admin Access/Features, and the ability to export a saved schedule. Both units were left untested due to time constraints leading to neither feature being implemented yet, however both are important with regards to the end goal of the application: Proving an easy to use experience, and a distinction between different types of users and their unique features.

Admin Access / Features

Admins would be a separate type of user from a student: An admin would be given the ability to not only modify a section’s capacity, but also view and edit any student’s current course list. Upon selecting a given course from their administrative list, admins would be given the option to change said course’s (as well its sections’) size capacity. This could be tested through trial (i.e.: Black-box testing) and by inputting various values in the section’s capacity is observing not only whether a change takes place, but also whether the value entered is a legal one (i.e.: No negative values). This is important to test to due to fact that courses and sections will natural change capacity from time to time, thus having the ability to manually do so is imperative.

Additionally, admins are given the ability to change any student’s current schedule. This is required as an administrator must be able to move students between courses and sections should the student come to them for help, or if they’re required to move certain individual due to classroom size limitations. This is done by adding a given student to a queue; a queue that may only contain one individual at a time. Once added, the administrator will be given the ability to view said students schedule, course list, and account information. From there, the process of adding and removing courses, as well as changing sections, is identical to the process used by the student. Once the necessary changes have been made, the student will be removed from the queue, allowing the admin to modify a different student. This can be tested, similarly to the previous step, with black box testing. By providing certain inputs to the administrative features, the results can be observed and the necessary modifications can be made. All administrative features are important due to the differentiation it provides between user types, and the unique role it plays in comparison to a normal student. Should they not be tested properly, students may be given administrative access, or administrators may be given none at all, thus ruining the differentiation created.

Export Saved Schedule

This feature is relevant due to the convenience it provides the students using The Force. By allowing their schedules to be exported, students are given the option to easily access their schedule without needing to go through the application (as well as print said schedule, providing them with a physical copy). As it is another requirement, this too would be tested through a black box, and should a properly exported file/image be created on request, the feature will be deemed successful. It is important to test due to the fact that, should it fail to work properly, user satisfaction would decrease as a result, and ease of access for the student’s schedule would be reduced. As the entire purpose of The Force is to create student schedules (primarily), it stands that once created, these schedules should be made easily accessible to those who made them.

1.2 Test Cases

1.2.1 Unit Testing

PHPUnit unit is a tool which exists to make unit testing easier and more precise for PHP.

The tools included are in the form of Test stubs and mocks. These 2 object have some of their components replace by fake ones. For example, an interior method can be replace with a fake method. In general, the stubs are allow us to write and run repeatable unit tests so that they last until the end of the life cycle of the system. The following tests presented used a branch and input/output analysis at the design phase of the unit tests.

In Unit Test 1 – Login, Mocks and Stubs are used. Authenticate creates a mock from UserID and creates a stub for the login method derived from UserID

Unit Case 1 - Login

With respect to the unit “Login”, two test cases were analyzed:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | *Login* Test Cases | |  |  |
| Test Case | | **Description** | **Input provided** | **Expected Output** |
| 1 | | Email and password combination provided is invalid | Email = “test.com”  Password = “passw” | User would be notified of the error, and the account would not signed in |
| 2 | | Email and password combination provided are valid | Email = “test@case.com”  Password = “password” | User would be signed in and redirected to main page |

For a detailed list of all possible test cases related to the requirements of “Login” (i.e.: The requirement that makes up this unit), refer to section 1.2.2. The results of the testing can be seen below. These two test cases were derived through grouping, and stubs were used in order to create mock student account (and other testing components), while a driver was created to simulate a new completed account and authentication of its values was allowed as a result. In order to simplify these two test cases, only two test functions were made: Login1 and Login2. Login1 would always return false (signifying an invalid email/password combination), while Login2 would always return true. In this way, though several different inputs could realistically be used, the results were simplified to two categories: Valid and Invalid. This simplified the act of testing and allowed the unit to be easily tested as a whole

<?php

require\_once('/../protected/layout/createAccount.php');

class LoginFormTest extends PHPUnit\_Framework\_TestCase

{

protected $driver;

//Set up the test driver

protected function setUp()

{

driver = new LoginForm;

driver->email = "test@case.com";

driver->password = "password";

}

Protected function tearDown()

{

}

public function Rules()

{

//Define the rules for the test stubs

$stub = array(array('username, password', 'required'),

array('password', 'authenticate'));

$results = driver->rules();

assertEquals($stub,$results);

}

public function Authenticate()

{

$stub = getMockBuilder('UserID')->getMock();

$stub->expects(any())

->method('authenticate')

->will(returnValue('true'));

assertTrue(driver->authenticate());

}

public function Login1()

{

$stub = getMockBuilder('LoginForm')->getMock();;

$stub->expects(any())

->method('login')

->will(returnValue('false'));

$this->assertEquals('false', $stub->login());

}

public function Login2()

{

$stub = getMock('LoginForm');

$stub->expects(any())

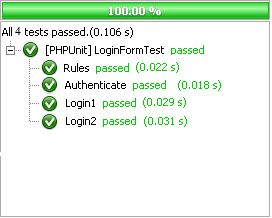
->method('login')

->will(returnValue('true'));

assertEquals('true', $stub->login());

} }?>

Unit Case 1 Results



Unit Case 2 – Create Student Account

With respect to the unit “Create Student Account”, two test cases were analyzed:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Create Student Account Test Cases | |  |  |
| Test Case | | **Description** | **Input provided** | **Expected Output** |
| 1 | | Email and password combination provided is invalid | Email = “test.com”  Password = “passw” | User would be notified of their error, and the account would not be created |
| 2 | | Email and password combination provided are valid | Email = “test@case.com”  Password = “password” | Account would be created, user would be redirected to main page |

For a detailed list of all possible test cases related to the requirements of “Requesting a New Account” (i.e.: The requirement that makes up this unit), refer to section 1.2.2. It is important to note that a test case was not created for when first or last names are not provided. As both fields are required, the result will never stray from simply asking for both again should they not be present (seen in the test stubs/driver code, as both fields are required). The results of the testing can be seen below. These two test cases were derived through grouping, and stubs were used in order to create mock student account (and other testing components), while a driver was created to simulate a new completed account and authentication of its values was allowed as a result. In order to simplify these two test cases, only two test functions were made: Create1 and Create2. Create1 would always return false (signifying an invalid email/password combination), while Create2 would always return true. In this way, though several different inputs could realistically be used, the results were simplified to two categories: Valid and Invalid. This simplified the act of testing and allowed the unit to be easily tested as a whole

<?php

require\_once('/../protected/layout/createAccount.php');

class createAccountTest extends PHPUNIT\_Framework\_TestCase

{

protected $driver;

//Set up the test driver

protected function setUp()

{

$driver = new createAccount;

$driver->firstName = "test";

$driver->lastName = "case";

$driver->email = "test@case.com";

$driver->password = "password";

}

protected function tearDown()

{

}

public function RulesT()

{

//Define the rules for the test stubs

$stub = array(array('firstName, lastName', 'required'),

array('email', 'authenticate'),

array('password','authenticate'));

$actual = $driver->createRules();

assertEquals($stub, $actual, "Values are not equal");

}

public function AuthenticateT()

{

$stub = getMockBuilder('studentAccount')->getMock();

$stub->expects(any())

->method('authenticate')

->will(returnValue('true'));

assertTrue($driver->authenticate());

}

public function Create1()

{

$stub = getMockBuilder('createAccount')->getMock();

$stub->expects(any())

->method('create')

->will(returnValue('false'));

//If not equal, print valid

assertEquals('false', $stub->create(), "Valid");

}

public function Create2()

{

$stub = getMockBuilder('createAccount')->getMock();

$stub->expects(any())

->method('create')

->will(returnValue('true'));

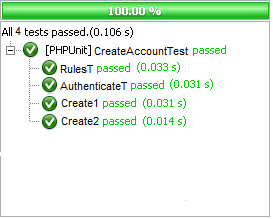
//If not equal, print invalid

assertEquals('true', $stub->create(), "Invalid");

}

} ?>

Unit Case 2 Results



1.2.2 Requirements Testing

The test cases for all tested requirements presented in section 1.1.1. can be found below. These requirements are based upon the use cases highlighted in deliverable 1 and 2. Each requirement includes numbered test cases, a description of each test case, an example of possible input which could be provided (as not every situation can be addressed for most requirements, thus partition testing is applied and test values are used to represent larger groups of examples) for said test case, the expected output, and whether each output was actually achieved. It is important to note that all test cases with failures are considered to be top priorities, and thus are primary focuses with regards to the end product of the application.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UC01 | Login |  |  |  |
| Test Case | **Description** | **Input provided** | **Expected Output** | **Result** |
| 1.1 | User inputs valid email and password combination | Email = “test@case.com”  Password = “password” | User is redirected to main page | PASS |
| 1.2 | User inputs an invalid email and password combination | Email = “test@case.com”  Password = “password1” | Empty email and password fields. User is not redirected | PASS |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UC02 | Logout |  |  |  |
| Test Case | **Description** | **Input provided** | **Expected Output** | **Result** |
| 2.1 | User is not logged in and attempts to log out | User attempts to press Logout button | User is on login page, and thus no logout feature is provided | PASS |
| 2.2 | User is logged in and attempts to logout | User presses Logout button in navigation bar | User is redirected to login page, without access to any other feature | PASS |
|  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UC03 | View / Edit Information |  |  |  |
| Test Case | **Description** | **Input provided** | **Expected Output** | **Result** |
| 3.1 | Student wishes to view their account | “Account” button is pressed | User is redirected from schedule/main page to Account page | PASS |
| 3.2 | User enters a new password and confirms their request | newPass = “password1”  confirm = “password1” | Message appears confirming user’s actions | FAIL |
| 3.3 | Administrator views a student’s profile | Student number is provided.  Example: “12345678” | Administrator is redirected to student’s account page | FAIL |

**Note that when adding a course (UC05), the case of have “co-requisites” completed but not prerequisites does not exist, as all courses in software engineering that require a prerequisite also allows the course to be completed concurrently. As such, this situation is not relevant.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UC05 | Add Course |  |  |  |
| Test Case | **Description** | **Input provided** | **Expected Output** | **Result** |
| 5.1 | Student adds a course they already have in their schedule | Student selects “COMP 232”, while COMP 232 is already in their schedule | No changes should occur | PASS |
| 5.2 | User adds a course they do not have the prerequisites, nor co-requisites, for | Student adds “SOEN 287” before completing “COMP 248” | Inform user of error (lack of pre/co-requisite) and do not add course to schedule | FAIL |
| 5.3 | User adds a course they have completed the necessary prerequisites for, but not co-requisites | Student has not yet added “SOEN 343”, but adds “SOEN 345” to their schedule and generates | Inform user of error (lack of co-requisite) and do not add course to schedule | FAIL |
| 5.4 | User adds a course they have previously completed (no need to retake the course) | Previously completed “COMP 248”, add to schedule once again | Inform user of error (course completed) and do not add to schedule | FAIL |
| 5.5 | User adds a course which conflicts with a course already in the schedule | “COMP 248” times overlap with “SOEN 228”, but the student adds “COMP 248” to their schedule anyways | Inform user of error (time conflict between courses) and do not add course to schedule | FAIL |
| 5.6 | User adds a course which is currently full (either a course or section) | “COMP 248” is already full, but is added to the schedule anyways | Inform user of error (course/section is full) and do not add course to schedule | FAIL |
| 5.7 | User adds a course with all the necessary prerequisites completed, no time conflicts, and space in the class remaining | “COMP 248” is added to the schedule, and all conditions are satisfied | Course and section is added to the student’s current schedule | PASS |
| 5.8 | Student adds a course after indicating preferred personal constraints (days off, between two times, etc…), assuming no other conflicts | Student selects “COMP 248” after indicating they would like no courses between 8:00 AM and 10:00 AM | Course and section would be added to the student’s current schedule | FAIL |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UC06 | Change Course Section |  |  |  |
| Test Case | **Description** | **Input provided** | **Expected Output** | **Result** |
| 6.1 | Student selects a course or section from within their schedule | Student clicks on “COMP 248” from within their schedule | Popup window appears containing alternate sections and their times | FAIL |
| 6.2 | Student selects a new section from the popup, but this conflicts with another course in the schedule | Student selects section “AA” for “COMP 248”, but this conflicts with “COMP 232” | Inform user of error (time conflict) and section change does not occur | FAIL |
| 6.3 | Student selects a new section from popup, but this section is full | Student selects section “AA” for “COMP 248”, but section “AA” is full | Inform user of error (section size full) and section change does not occur | FAIL |
| 6.4 | Student selects a new section from popup, and no issues are associated with this section | Student selects section “AA” for “COMP 248” | Section change occurs. Both course list and the student’s schedule are updated | FAIL |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UC07 | Remove Course |  |  |  |
| Test Case | **Description** | **Input provided** | **Expected Output** | **Result** |
| 7.1 | Student tries to removr course when no courses are saved | Student presses “Remove” button | No “Remove Course” button is present | PASS |
| 7.2 | Student chooses to drop a course, and confirms their decision | Student finds the course they wish to remove in their schedule, clicks it, and selects “Remove Course” | The course is removed from the course list, as well as from the saved schedule | FAIL |
| 7.3 | Student chooses to drop a course, and cancels their decision | Student finds the course they wish to remove in their schedule, clicks it, and cancels their decision | The course will remain in the course list and schedule | FAIL |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UC08 | Generate Schedule |  |  |  |
| Test Case | **Description** | **Input provided** | **Expected Output** | **Result** |
| 8.1 | Student auto generates a schedule for the current semester | Student presses “Auto Generate Schedule” button | A popup will appear containing an auto generate schedule, based on the next courses in said student’s sequence | FAIL |
| 8.2 | Student dismisses the option created automatically | “Cancel” button is pressed on popup window | Schedule will not be saved and user will remain on main page | FAIL |
| 8.3 | Student confirms the schedule displayed | “Accept” button is pressed on popup window | Schedule will be saved and will now be featured on the main page, next to a course list | FAIL |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UC11 | View Saved Schedule |  |  |  |
| Test Case | **Description** | **Input provided** | **Expected Output** | **Result** |
| 11.1 | Student views their schedule | Student signs in, makes an account, or presses “Schedule” from the “Accounts” page. | Student is redirected to the main / schedule page and their current saved schedule is displayed. | PASS |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UC16 | Request New Account |  |  |  |
| Test Case | **Description** | **Input(s) provided** | **Expected Output** | **Result** |
| 16.1 | Create Account option is chosen | User presses “Create Account” button | Popup page with all required fields appears | PASS |
| 16.2 | User leaves any of the 4 fields blank (First name, Last name, Email, Password) | Any blank field will result in the same expected output.  First = “Test”  Last = “Case”  Email = “”  Password = “password” | Popup window remains open. Field with error is highlighted in red in order to notify user. | PASS |
| 16.3 | Email does not follow the format “xxx@yyy.zzz” | 1. @test.com 2. testcase.com 3. test@case | Popup window remains open. Email field is highlighted in red. Message appears below field notifying user of invalid entry | FAIL |
| 16.4 | Email follows the format “xxx@yyy.zzz” | Email = “test@case.com” | No error message displayed. Assuming other fields are valid on submission, registration is completed | PASS |
| 16.5 | Password is less than 6 characters | Password = “passw” | Popup window remains open. Password field is highlighted in red. Message appears below field notifying user of invalid entry | PASS |
| 16.6 | Password length is equal to or greater than 6 characters | Password = “password” | No error message displayed. Assuming other fields are valid on submission, registration is completed | PASS |
| 16.7 | Previously completed courses are marked (checkboxes) | Checkbox indicating “COMP232” is checked | Assuming successful registration, marked courses can be seen in “Accounts” page | FAIL |
| 16.8 | All fields are filled with valid entries | First = “test”  Last = “case”  Email = “test@case.com”  Password = “password” | Upon submission, user is redirected to main schedule page. | PASS |

1.2.3. Stress Testing

In the case of extreme situations that are beyond our control, the system might encounter decline in performance with regards to software. This may be related to the ability of the server to handle heavy networking loads due the cap on capacity.

Under conditions such as high latency and a large number of simultaneous website connections, we would experience clear degradation of service. The system would no longer be able to meet outlined requirements such as response time, uptime, and scalability. Testing all of the conditions would be possible if and only if the system was put onto a live environment.

Based on experience and through knowledge obtained by observing other systems, we have concluded that time periods where the system could face the conditions mentioned are during the scheduling periods of the academic year. Since these periods are known and occur around the same time every year, it would be possible to act accordingly and prevent the issues.

One possibility would be to dedicate extra bandwidth to the site during the scheduling period of the academic year. This could be done by purchasing or renting additional servers, or by changing TCP/IP protocol to better manage bandwidth used by an individual person.

To test for performance and reliability under conditions of extreme load, it is possible to setup a testing environment where a script can be run that was designed to bombard the system with a variety of simultaneous function calls. These function calls could include login/logout, register course, and/or any of the other implemented features outlined above. The script could also be run on multiple computers at once to test for higher traffic.

1.2.4. Security Testing

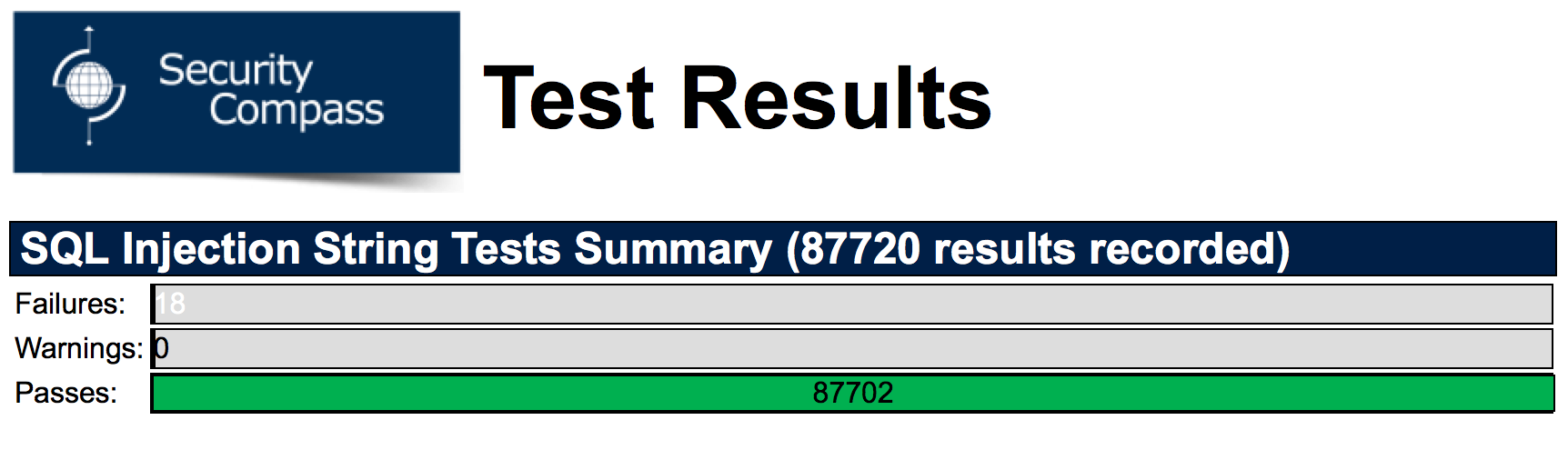
**SQL Injection**

SQL injection is a technique by which malicious SQL statements are inserted into an entry field in order to trick the system into giving more information than would be normally available. By exploiting wording used in SQL statements, security vulnerabilities can be exploited and incredible damage can be done to an application, thus it is imperative that this threat be tested against.

In order to test for these vulnerabilities, a FireFox Extenstion called SQL Inject-Me was used. By submitting HTML forms with substituted values (i.e.: Strings representing what would be used during an SQL Injection attack), the tool allows for the inspection of said vulnerabilities. Once escape strings have been sent to the database through the form fields, the extension then looks for database error messages that have been outputted into the rendered HTML of the page. Despite all this, the tool does not compromise the security of the application, as it simply looks for entry points to attack through (no actual attacks are done).

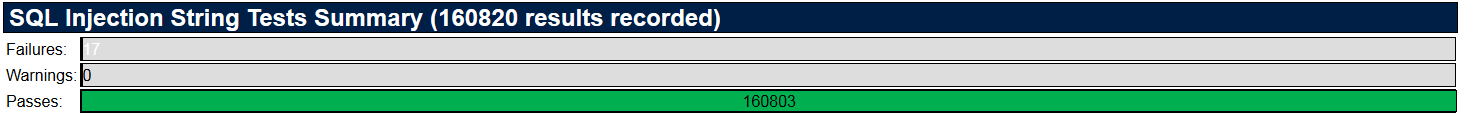
The SQL Injection tool was run on 3 of The Force’s pages: It’s login page (including both logging in and creating an account), the main / schedule page, and the user account page. Below are the results. As the results (when errors were found) and number of form fields can be quite extensive at times, only the overall results are shown. To mitigate this, explanations for failures are provided when they occur.

1. Login Page

****

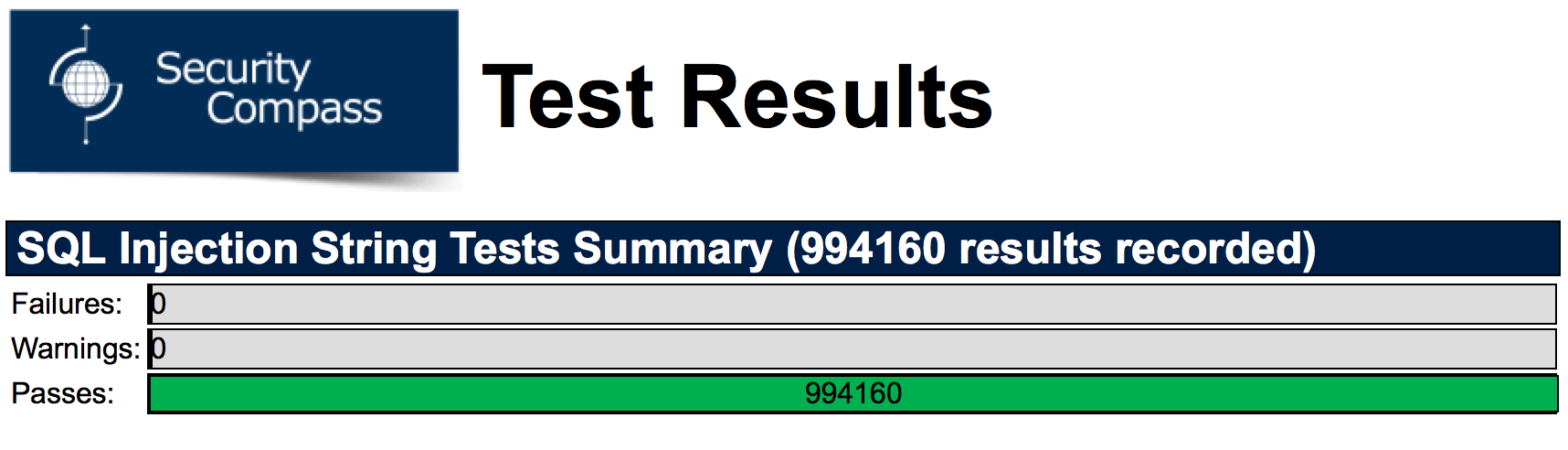
The only (18) failures experienced on the login page were associated with the email and password fields, thus extra precaution will be taken with respect to the final project to ensure that security is thoroughly provided for these two fields.

1. Main Page



On the main page, the extension detected 17 failures. However, these failed test cases were all encountered on the same field: The second “unnamed field” located at the bottom of the main page (at the time of this testing). This extra field is not relevant to the main functionality of the scheduler, and will be fixed to avoid security vulnerabilities once all revisions have been completed. Removing this field is expected to remove the failed cases with it.

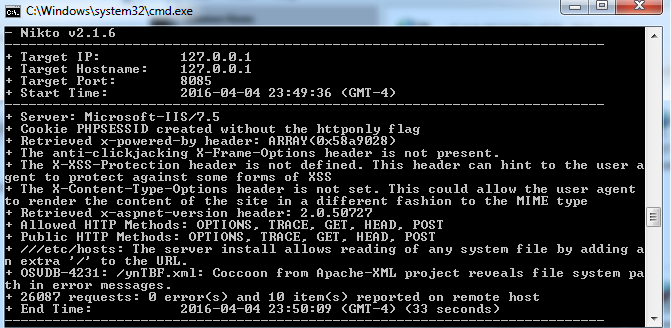
1. Account Page



No failures or warnings were detected on the accounts page, thus it can safely be considered secure from SQL Injection attacks.

**Nikto**

Nikto is an open source web server scanner, used to perform comprehensive tests against web servers from multiple items (e.g.: potentially dangerous files/programs, outdated server versions, and version specific server issues). As web server security is an ever-increasing issue, namely since Web Server vulnerabilities can be easily taken advantage of, it is imperative that the server used for The Force be tested for security. The Force is hosted on an IIS (Internet Information Services), which is a Microsoft-made extensible web server, and the results of running Nikto on our server can be found below.



The security tool failed to find any particularly dangerous errors, and thus the server security can be considered to be quite high.

**NMap**

NMap (or Network Mapper) is a security scanner used to discover hosts and services on a computer network. In addition, NMap can be used as a port scanner, to determine whether network ports are open, closed, filtered (i.e.: NMap can’t determine if the port is opened, making it frustrating for attackers), etc.… The main result of this scan can be seen below:



As can be seen above, several ports are currently open. While this makes accessibility higher (i.e.: For the team), open ports are simply paths to be attacked through. Since possible attackers wish to exploit open ports, it would be ideal to close (or at the very least filter) these ports once work on the project has completed. In doing so, the legitimate users (the team) will not be reprimanded, but possible attackers will no longer be able to utilize these open ports to target the network (as ports present an endpoint of communication in an operating system).

2. System Delivery

2.1 Installation Manual

The following will explain how you can go about installing a web-application service tool on your hosting server. *Note: This will only function on a Windows device. Ensure that all site files provided are stored in a specific folder (ex: C:\Websites\Generator) so they are easier to access.*

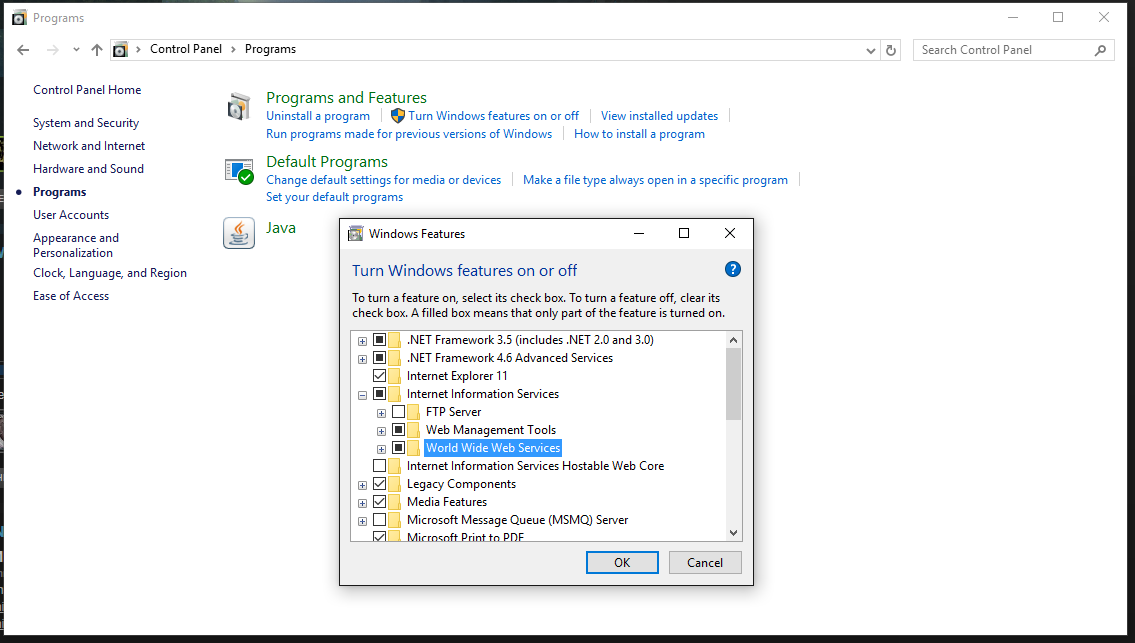
1. Open the Control Panel on your Windows device.

2. Select *Programs.*

3. Under *Programs and Features*, select *Turn Windows Features On or Off*.

4. Under *Internet Information Services*, check *Web Management Tools* and *World Wide Web Services*.

* This will install required files and the IIS manager application needed to host a website.

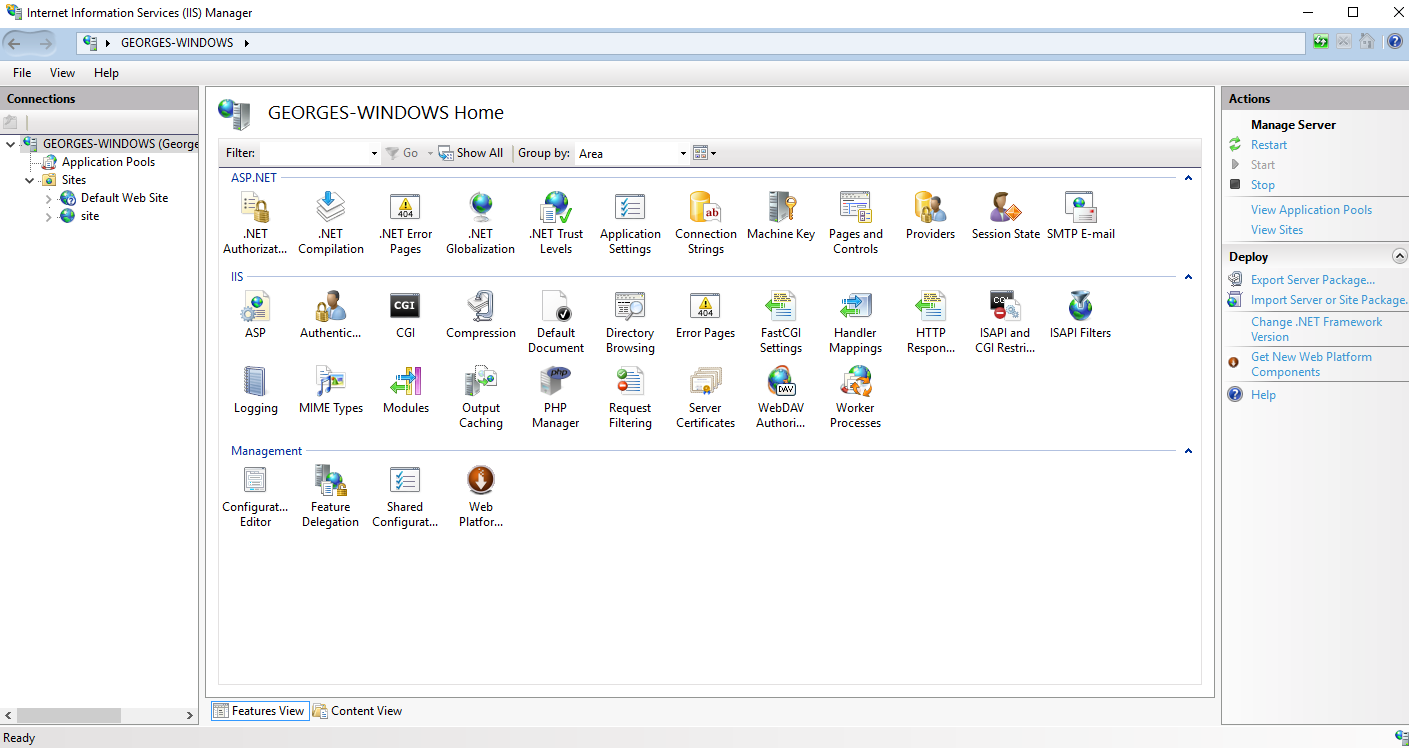


*Control Panel*

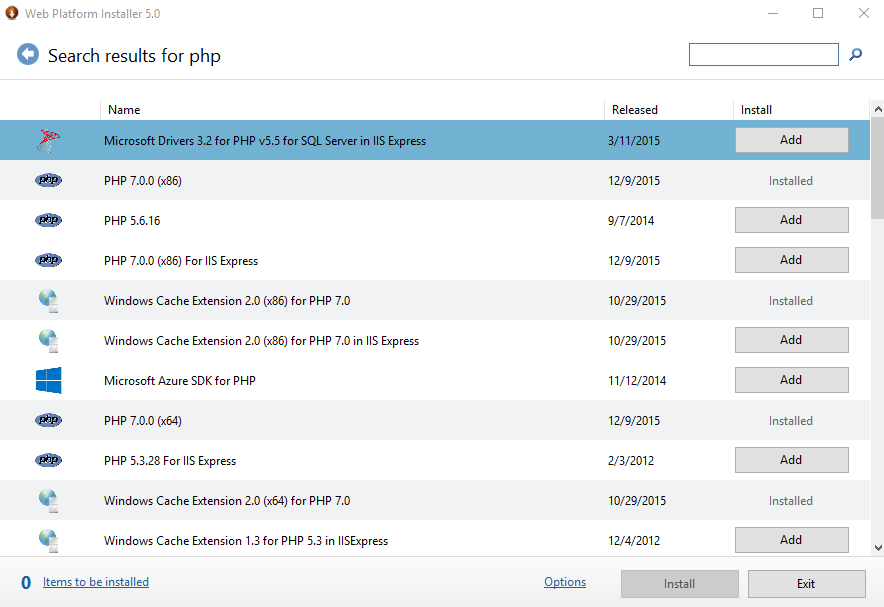
5. Download and install PHP for IIS from the following website: <https://php.iis.net/>

* This is what will allow you to run the PHP scripts from the websites

6. Open the IIS manager application installed in step 4.

7. Open *Web Platform Installer*

*IIS Manager*

8. Search *php* in the search bar. Find the version of PHP that was just installed and is now being used (likely PHP 7.0), and click *add* to install it to the IIS Manager Application.

*Installing PHP for IIS manager*

9. Return to the IIS manager default page. On the left hand side, right click your computer

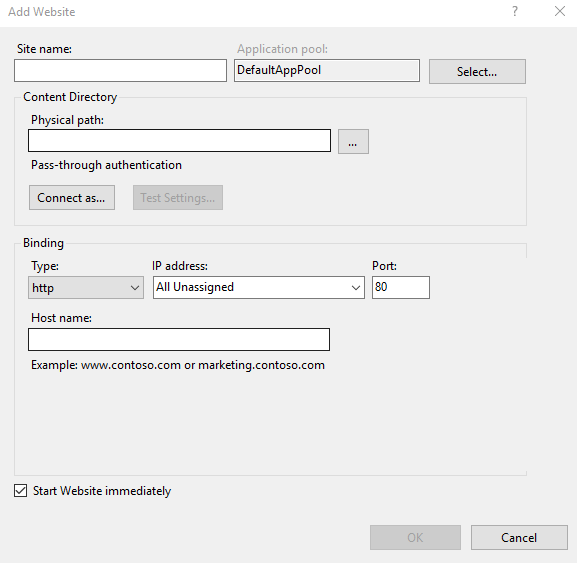
name. Select *Add Website*.

10. In the *Site name* field, enter your desired name for the website.

11. In the *Physical path* field, select the folder that contains your website files.

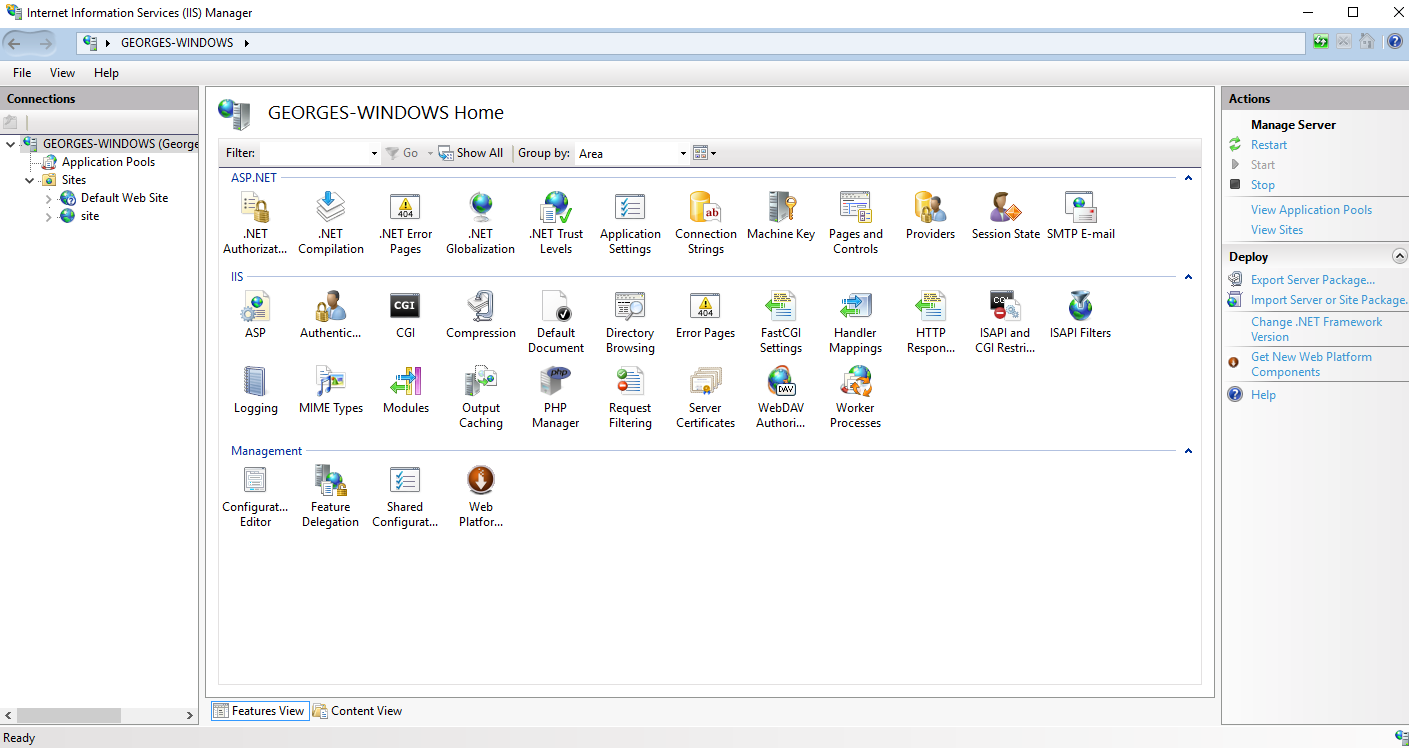
12. In the *Binding* section, leave *Port* as default. This value should be 80; if it is not, set it to

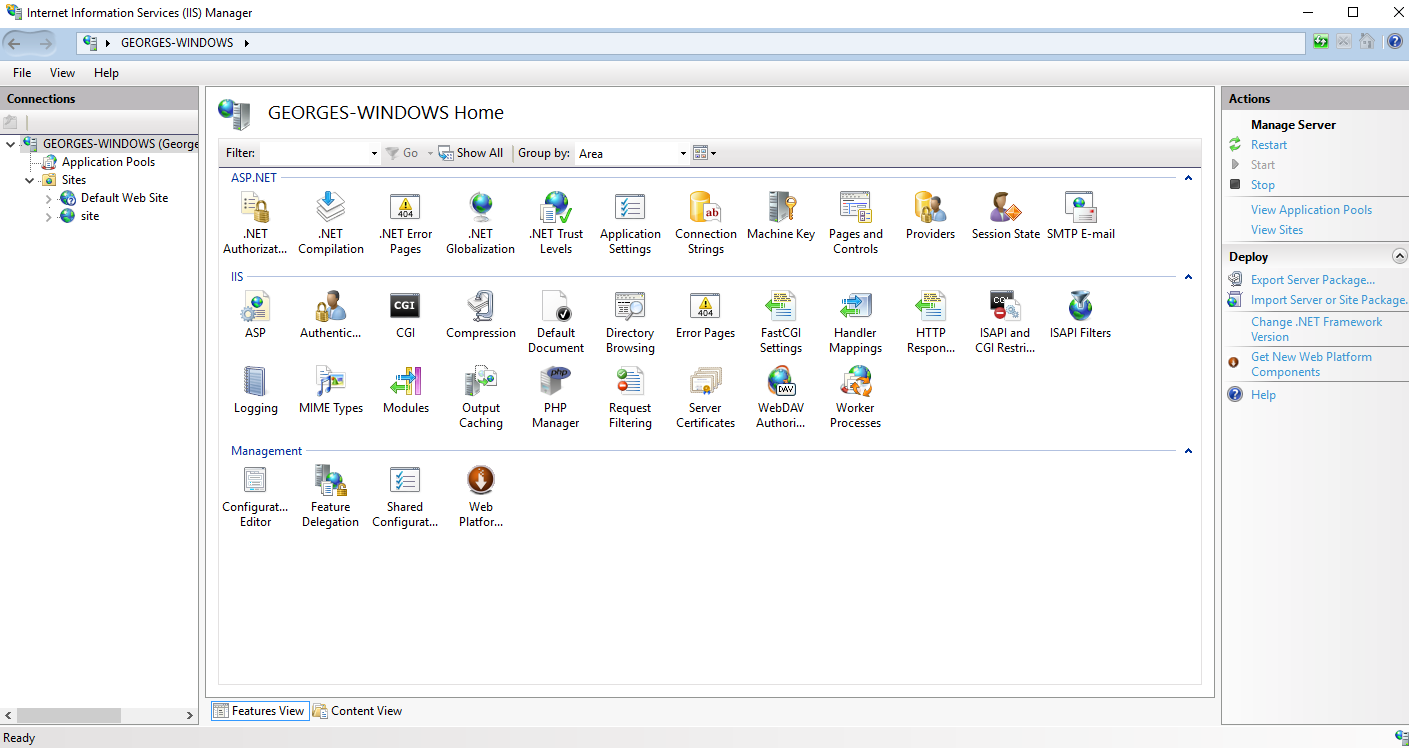
80 manually.

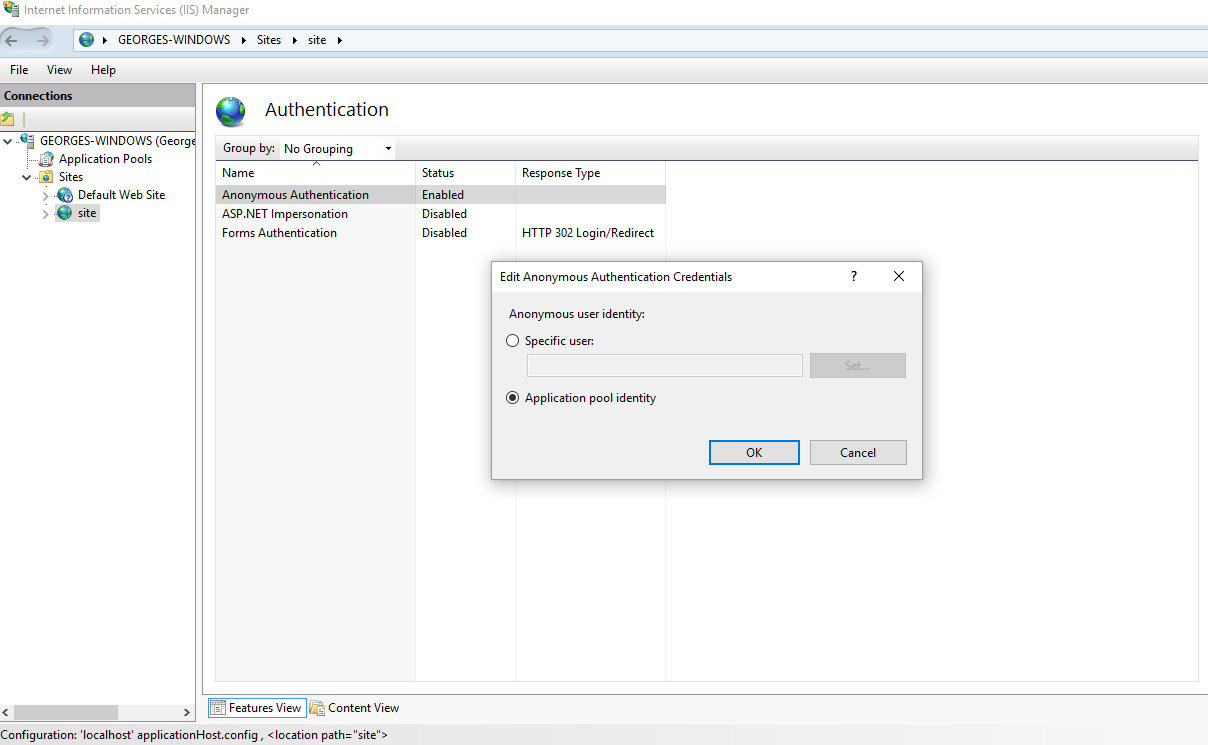


*Add Website*

13. Select *Default Document* and change it to *signin.php*.

* This will be the sign in page of the website you would like to host

14. Select *Authenticator* and ensure that *Application pool identity* is selected.

* This ensures you are given the sole administrator access to the system.

15. Download and install MySQL at the following link: <https://www.mysql.com/downloads/>

16. Select the scripts located in *dbscripts* in MySQL workbench and run them.

* This allows all SQL scripts to store information on the database of the server

Congratulations! Your website should now be up and running.

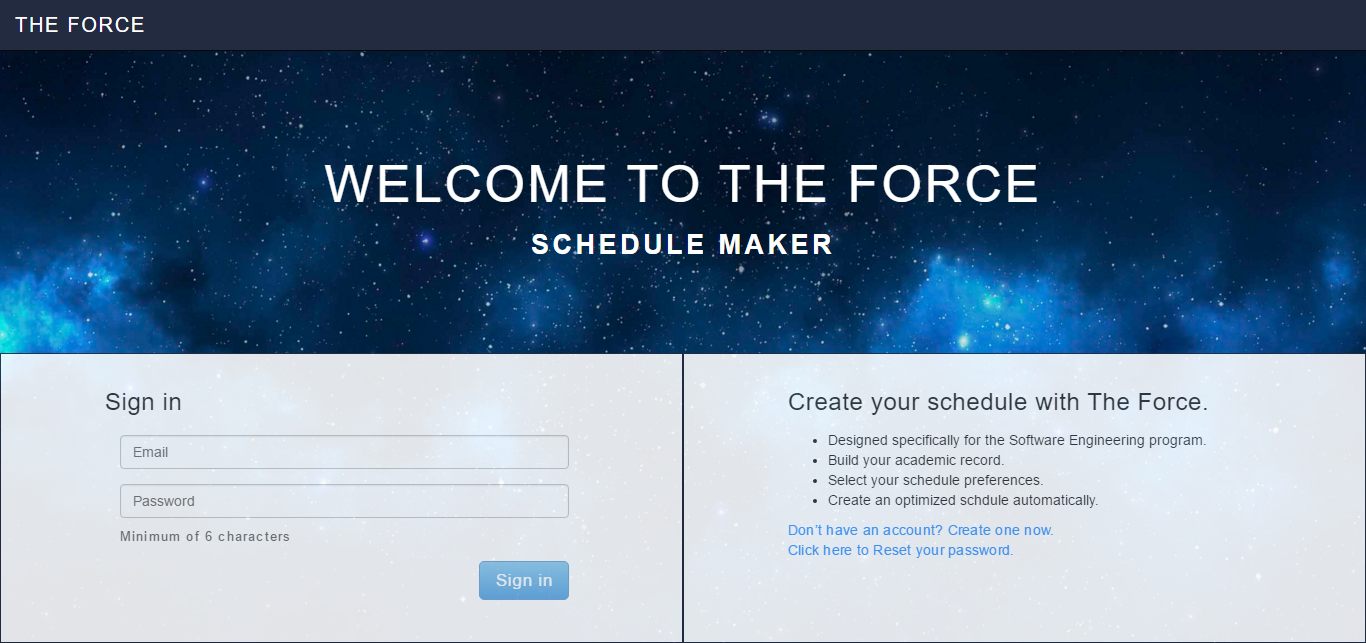
2.2 User’s Manual

2.2.1 Getting Started

The application is web based. Open your web browser of choice and navigate to the following:

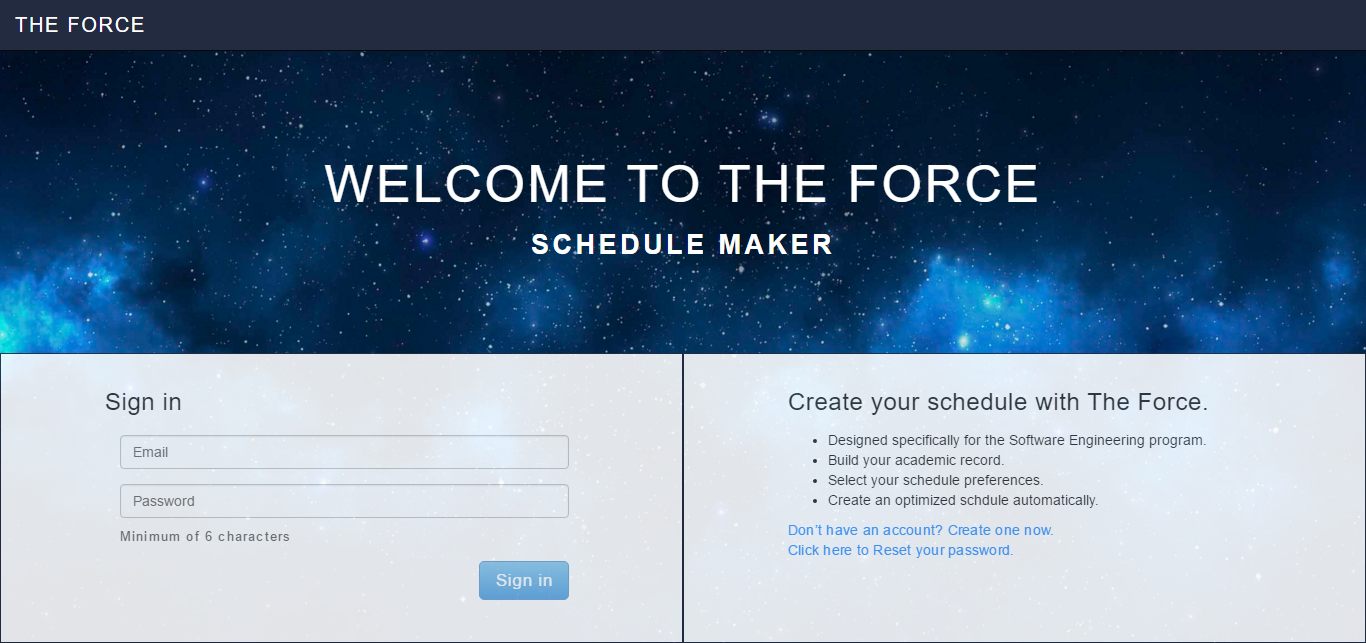
<http://wolfcall.ddns.net:8085>

You will be directed to the following Sign In page:

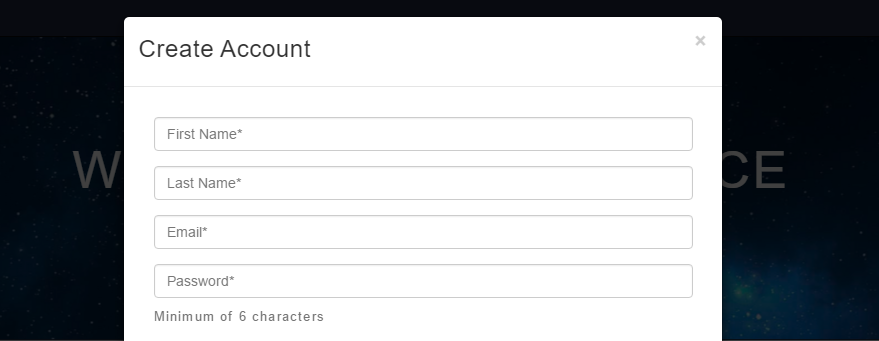


2.2.2 Creating an Account

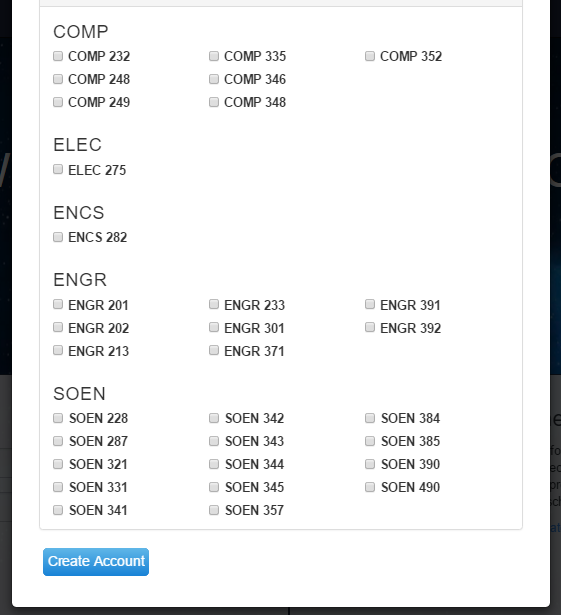
If you do not have an account, you can select the “Don’t have an account? Create one now” located by the red box.



Once selected, you will then be prompted to enter your personal information. This includes first name, last name, email and desired password (minimum 6 characters):



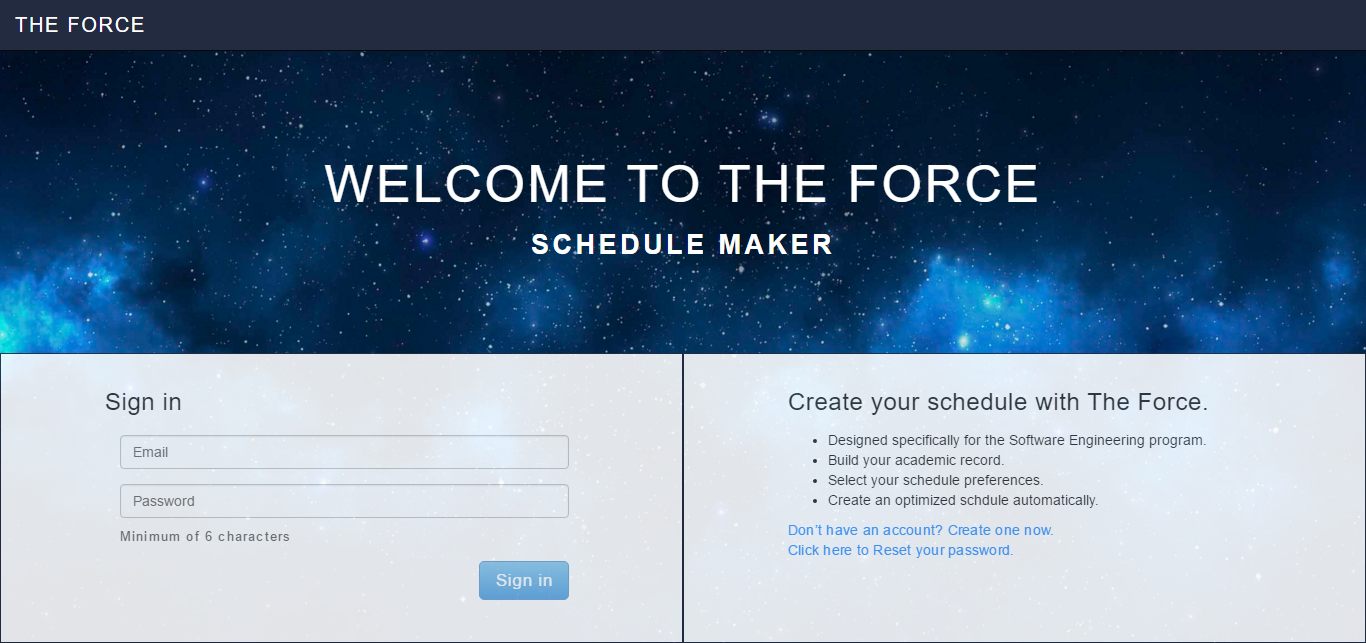
As well, you will need to check off the classes you have previously completed during your studies, so they can be saved as prerequisites for your account.



Once you select “Create Account”, your account information will be saved. Upon successful creation of your account, you will be redirected to the main page of the website (see later). If your account creation was not successful, verify if all fields are valid. Invalid fields will be highlighted in red.

2.2.3 Login

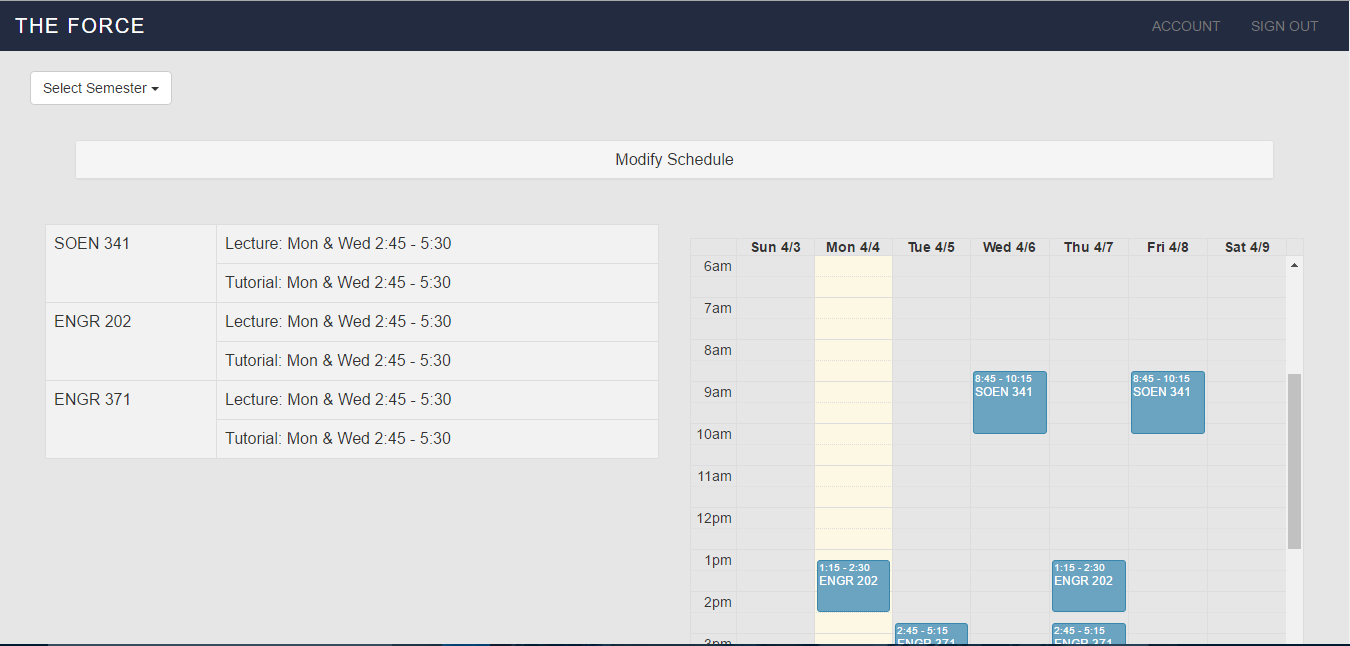
If you wish to log into your account, you will be required to fill in your email address and password information (shown by the red box).



Once the email and password combination is verified, you will be redirected to the main page of the website (see later). If your sign in was not successful, verify that all fields are valid. Invalid fields will be highlighted in red.

2.2.4 Logout

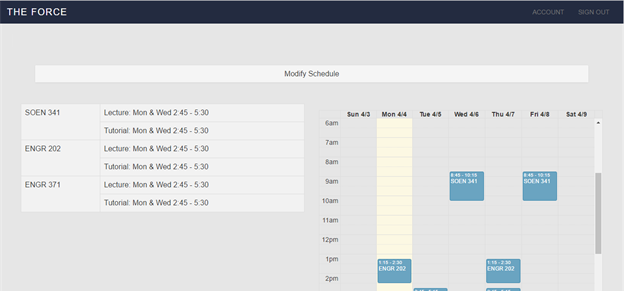
If you wish to log out of your account, you may select the “Sign Out” option in the top right corner of the screen that is shown by the red box.



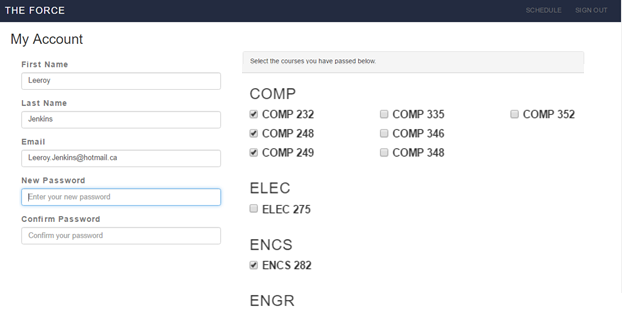
Once signed out, you will be redirected to the Sign In page.

2.2.5 View/Edit Profile

If you wish to edit your account information, you can select “Account” on the top right corner of the website.



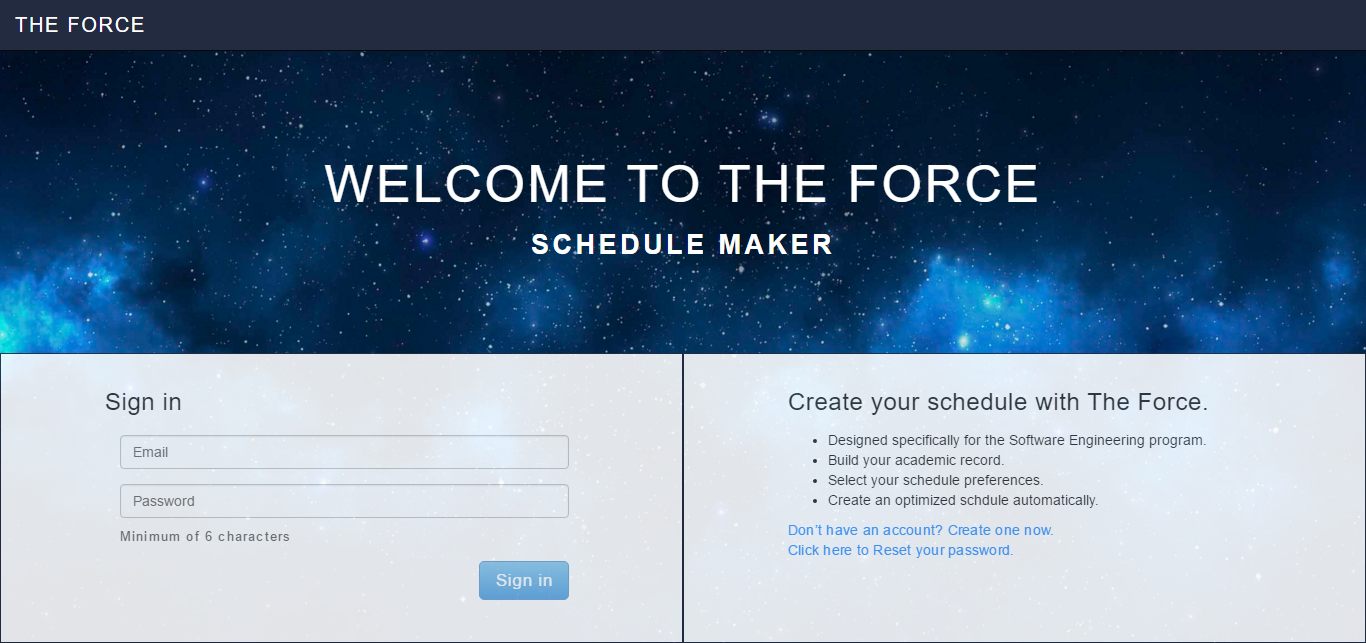
You will then be redirected to the My Account page. Here you will be able edit any of the information you previously entered such as first name, last name, email and courses previously taken. You also have the option to create a new password shown by the red box. You can always go back to the schedule page to view saved schedule by selected “Schedule” (shown in yellow).



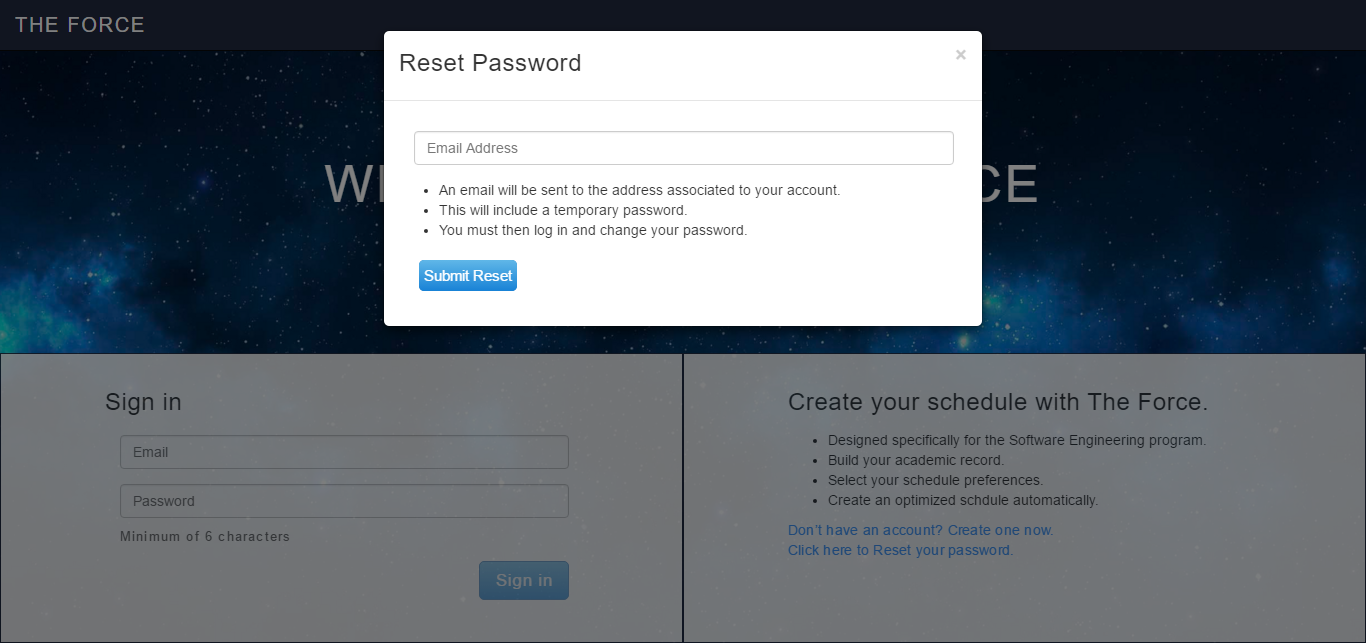
Once you select “Submit Changes” located lower on the page (not shown in the photo), these changes will be saved.

2.2.6 Reset Password

If you wish to reset the password of your account, then you may select “Click here to Reset your password” as shown by the red box.



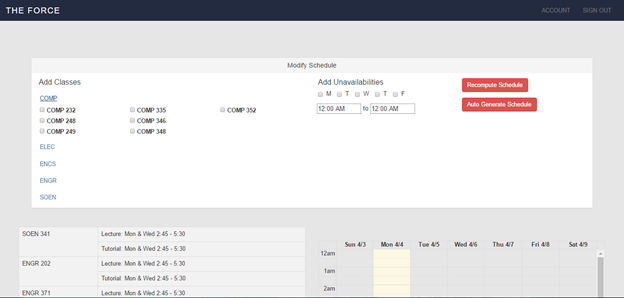
You will then be prompted to enter the email associated to your account. Once the email has been verified, a temporary password will be sent to you. You must then log in and change your password by going to the “Account” section described earlier.

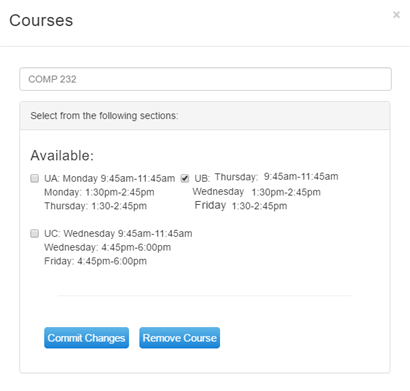


2.2.7 Add Course Manually

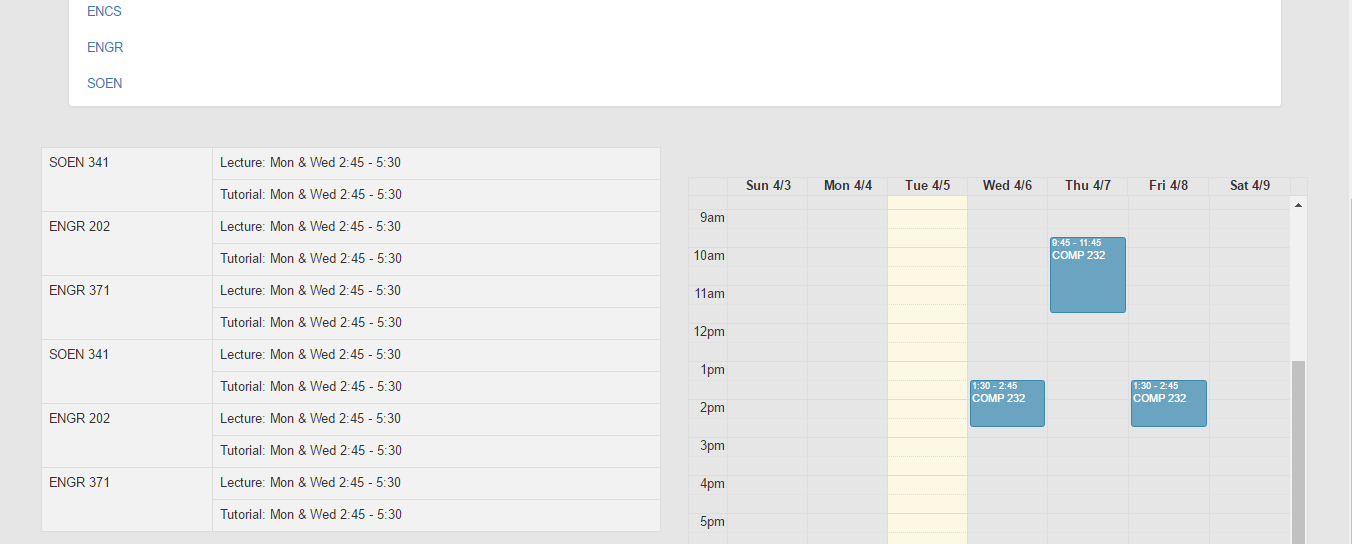
If you wish to add a course to your schedule, then you may select “Modify Schedule”

You will then see the following options. You can select the “COMP”, “ELEC”, “ENGR” or “SOEN” disciplines to expand a list of course (shown by the red box). Once a course is selected, you can specify unavailable times or days. Once you select “Recompute Schedule” (shown by the yellow box), the sections available will be shown and you can specify the one to add to the schedule.





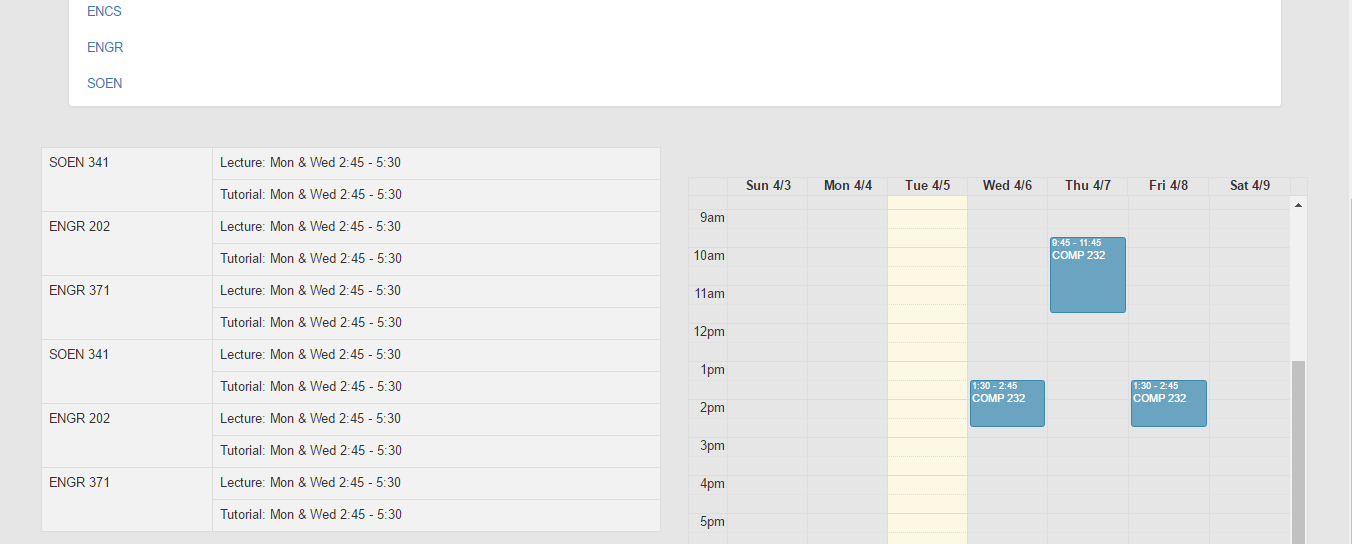
As an example, COMP 232 will then be displayed as follows:



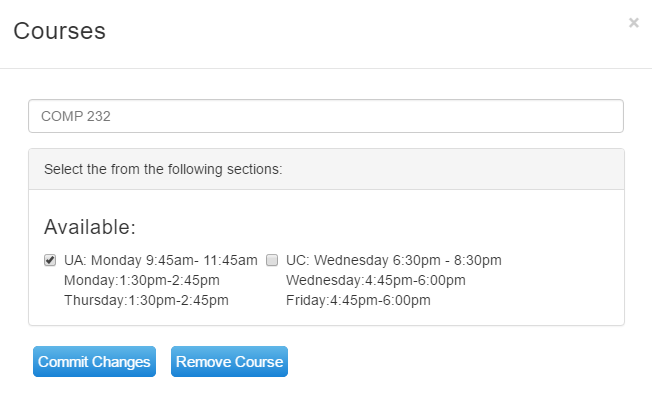
If there are any conflicts between the course you are trying to add and the courses already in your schedule, then an error message will be displayed. You must either choose another section for one of the classes in the conflict, or choose one for removal.

2.2.8 Change Course Section

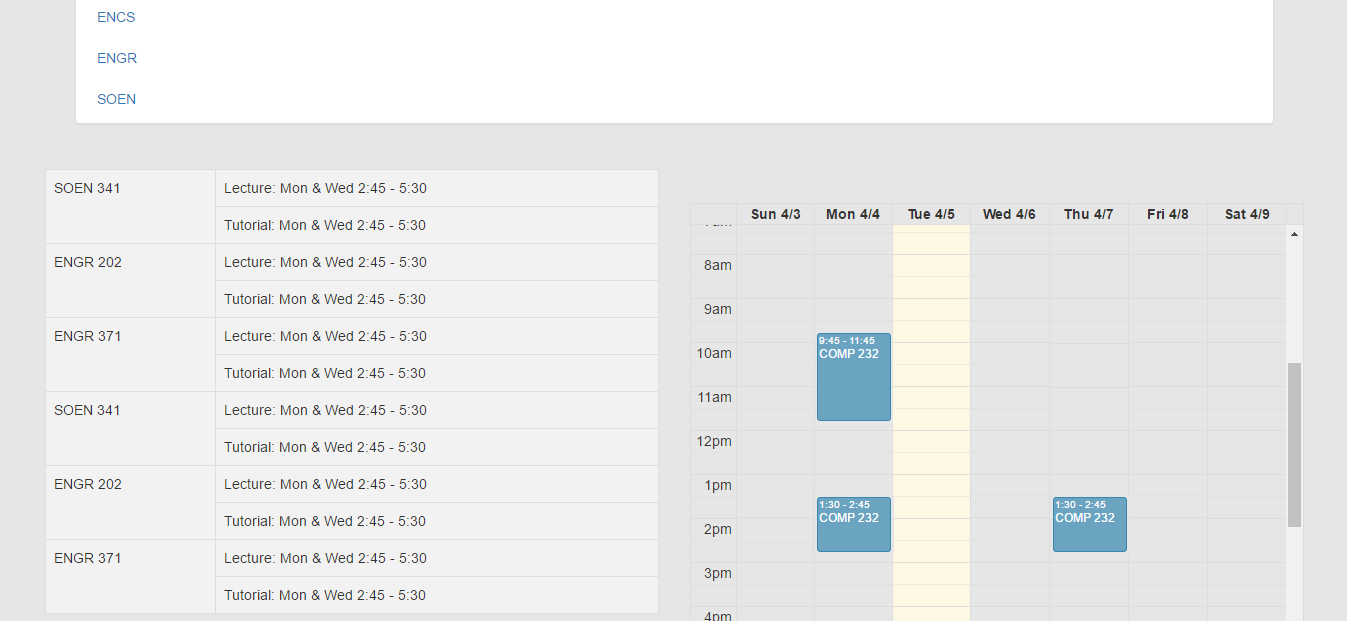
If you wish to change a section of a course you are currently enrolled in, you must first select the course from your schedule (shown by the red box). For this example, we will use the same course as mentioned in the last section: COMP 232.



Once COMP 232 is selected, a prompt will appear for you to choose from the remaining sections. When you pick the new section you would like to have, you may select “Commit Changes” (shown by the red box).

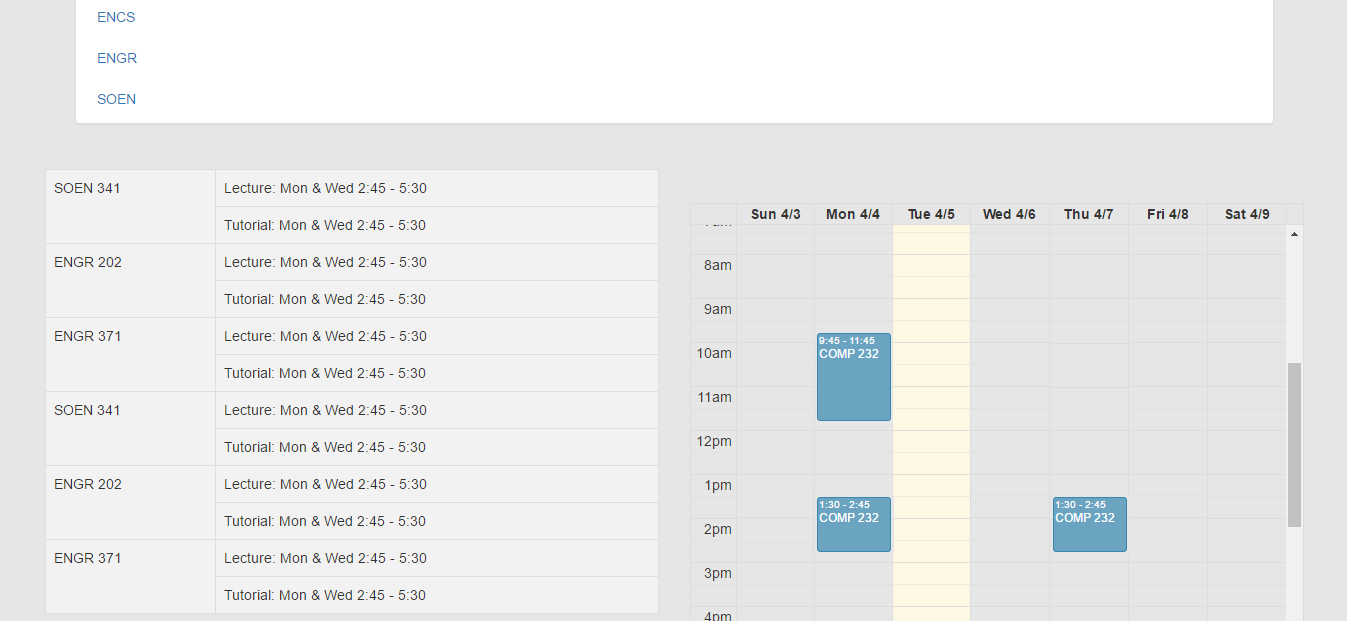


Afterwards, the new section will be displayed in your schedule as follows:

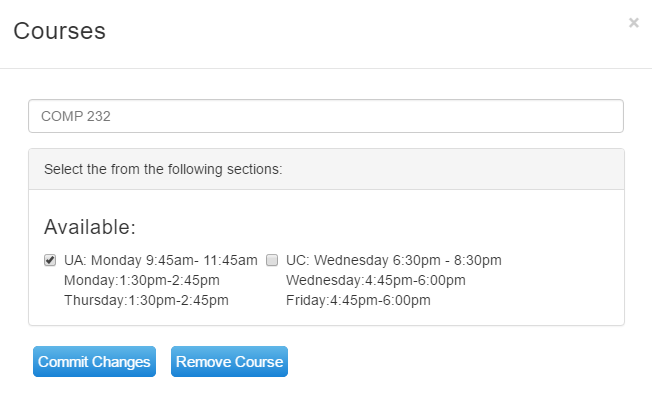


2.2.9 Remove Course

If you wish to remove a course from your schedule that you are currently enrolled in, you must first select the course from your schedule (shown by the red box). For this example, we will again use the same course as mentioned in the last section: COMP 232.



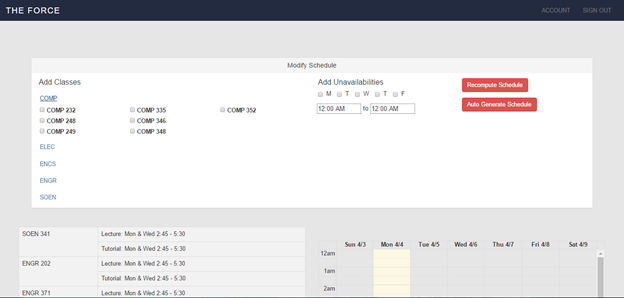
Once COMP 232 is selected, a prompt will appear for you to confirm the removal of this course. When you are certain that you will remove this course, you may select “Remove Course” (shown by the Red Box).



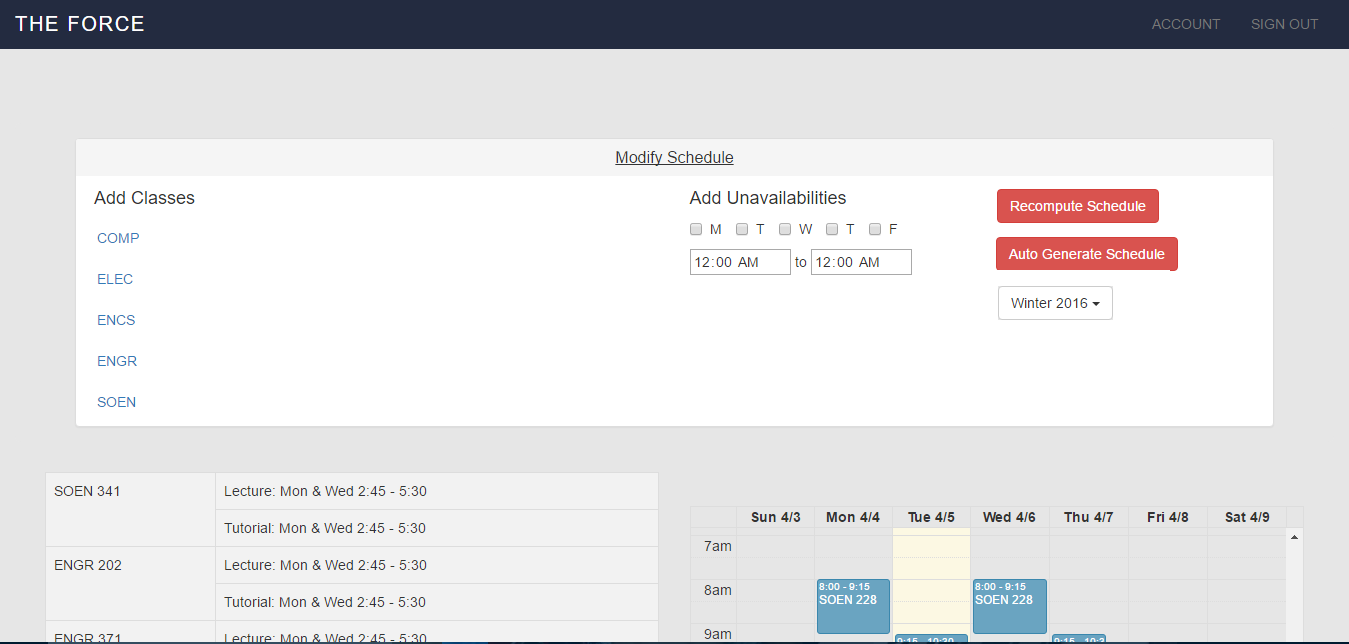
Afterwards, the course will be removed from your schedule.

2.2.10 Schedule Generation

If you wish to automatically generate your schedules for the remaining courses in your sequence, then you must first specify unavailable times or days. Once you select “Auto Generate Schedule” (shown by the yellow box), an acceptable set of schedules at random will be created by the system for the various semesters.



The first schedule will then be displayed, followed by a drop down menu for you to browse through upcoming semesters (shown by the red box).



If there are any conflicts between the courses auto-generated, then an error message will be displayed. If this is the case, you will need to contact your administrator, as your remaining courses conflict and prevent you from completing your sequence.

2.2.11 Export Schedule

Due to complications with implementation, this feature was not incorporated into the system yet. However, the following set of instructions remain valid and they will be updated to include screenshots when the system incorporates this feature.

If you wish to export one of your saved schedules to PDF, you may select “Export Schedule”. The system will then begin a download of the schedule to your computer in PDF format.

2.2.12 Modify Section Capacity (Administrator)

Due to complications with implementation, this feature was not incorporated into the system yet. However, the following set of instructions remain valid and they will be updated to include screenshots when the system incorporates this feature.

As an administrator, you are given more features than a regular user. On your main page, you are given the ability to search through all available sections for a specific course. if you would like to change the capacity of a certain section, you must select “Change Capacity” and enter the newly desired capacity. The system then changes the capacity of the section and you will be redirected to the main page of the website.

2.2.13 Add Student (Administrator)

Due to complications with implementation, this feature was not incorporated into the system yet. However, the following set of instructions remain valid and they will be updated to include screenshots when the system incorporates this feature.

Another Administrator privilege is the ability to edit any existing student’s schedule. To do this, as an Administrator you must first add the student to your queue by searching for their name. Once the student is found, the system will display any schedule they have created and give you the ability to scroll through various semesters through the use of a drop down menu.

\*\*\*\*\*\*Please note you can only have 1 student at a time in the queue.

Now that you can view any of the student’s schedules, you are free to modify them in any way you please, using features that were mentioned in previous sections (add course, remove course). These features are shared between regular user and Administrators. Any change in the schedule will result in the system displaying the new schedules.

2.2.14 Remove Student

Due to complications with implementation, this feature was not incorporated into the system yet. However, the following set of instructions remain valid and they will be updated to include screenshots when the system incorporates this feature.

As an administrator, once you are done editing a specified students’s schedule, you may remove him from your queue by selecting “Remove Student”. Once this is selected, you will be re-directed to the search page where you can search for another student if need be.

# 3. Final Cost Estimate

## Revised Project Estimate

This is a copy of the Project Estimates table from Deliverable 2

|  |  |  |  |
| --- | --- | --- | --- |
| 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 | 1 | 3 | 02-Feb |
| ER Diagram | 1 | 3 | 06-Feb |
| Architecture Programming | 20 | 90 | 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 | 472 | N/A |

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 | 40 | 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 | 20 | 75 | 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 | 1 | 3 | 02-Feb |
| ER Diagram | 1 | 3 | 06-Feb |
| Architecture Programming | 20 | 60 | 19-Feb |
| Scope of Project | 60 | 15 | 08-Feb |
| Test Cases | 12 | 25 | 28-Feb |
| Performance | 8 | 20 | 08-Mar |
| Design of UI / Aesthetics of Application | 10 | 15 | 12-Mar |
| Debugging | 10 | 45 | 22-Mar |
| Deliverable 4 : Final Report (Documentation) | 5 | 25 | 6-Apr |
| Total | 95 | 512 | N/A |

The changes to the Project Estimates table were made for the following reasons:

Deliverable 3 (Increased from 15 days, 30 hours 🡺 15 days, 40 hours)

* This deliverable proved to be more difficult than originally expected and was completed in a longer time frame.
* As a result we had to get more members to help complete the activities.

Implementation (Increased from 15 days, 35 hours 🡺 20 days, 75 hours)

* This task became very complex very quickly. A considerable amount of extra hours had to be allocated to this task because the team encountered serious configuration issues with the server hosting the website.

Architecture Programming (Decreased from 20 days, 90 hours 🡺 20 days, 60 hours)

* After the complications with the servers, we determined that we needed to work efficiently to complete this task. Again, with all the members of the team working, we were able to complete the task early.

Test Cases (Increased from 12 days, 15 hours 🡺 12 days, 25 hours)

* After starting the testing cases, we felt like we were not being thorough enough so we allocated more time for more tests.

Debugging (Increased from 10 days, 30 hours 🡺 10 days, 45 hours)

* Since more tests were done, more bugs were found so more time was taken in order to fix the errors.

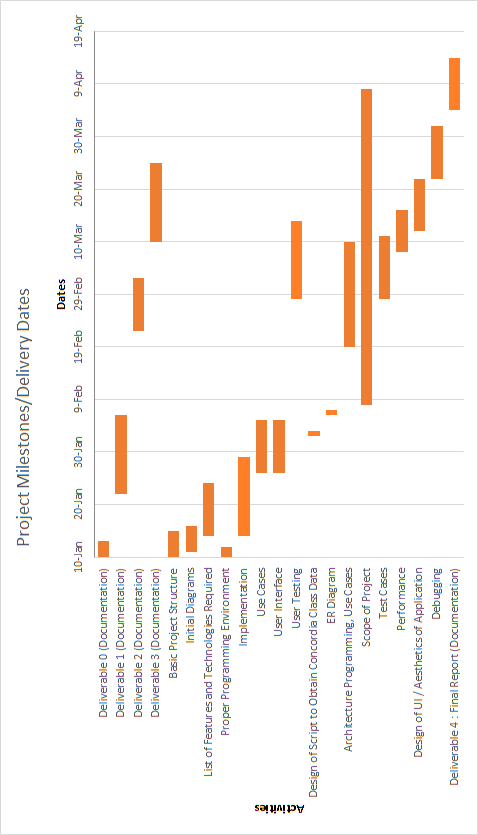
Deliverable 4 (Decreased from 10 days, 30 hours 🡺 5 days, 25 hours)

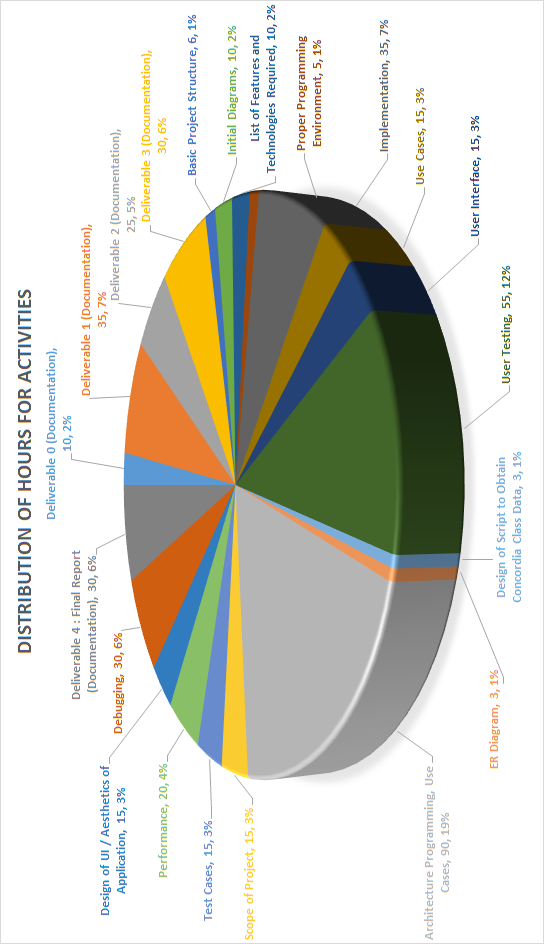
* Since editing was done as feedback was given for all the deliverables, the editing process for the Final Report is expected to be completed in less time than originally thought.

\*\*\*\* 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.

## 3.2 Final Schedule & Gantt Chart

This is a copy of the original Schedule & Gantt Chart. A pie chart was also included to show the initial distribution of hours for each task in comparison to the project as a whole.

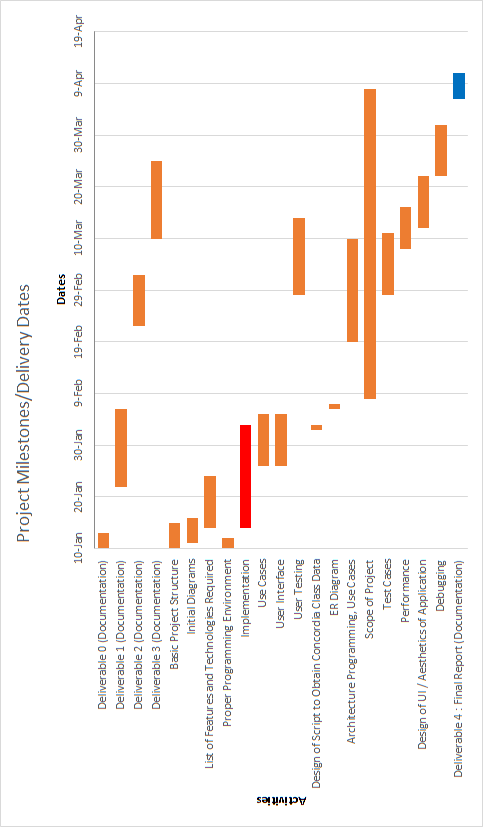


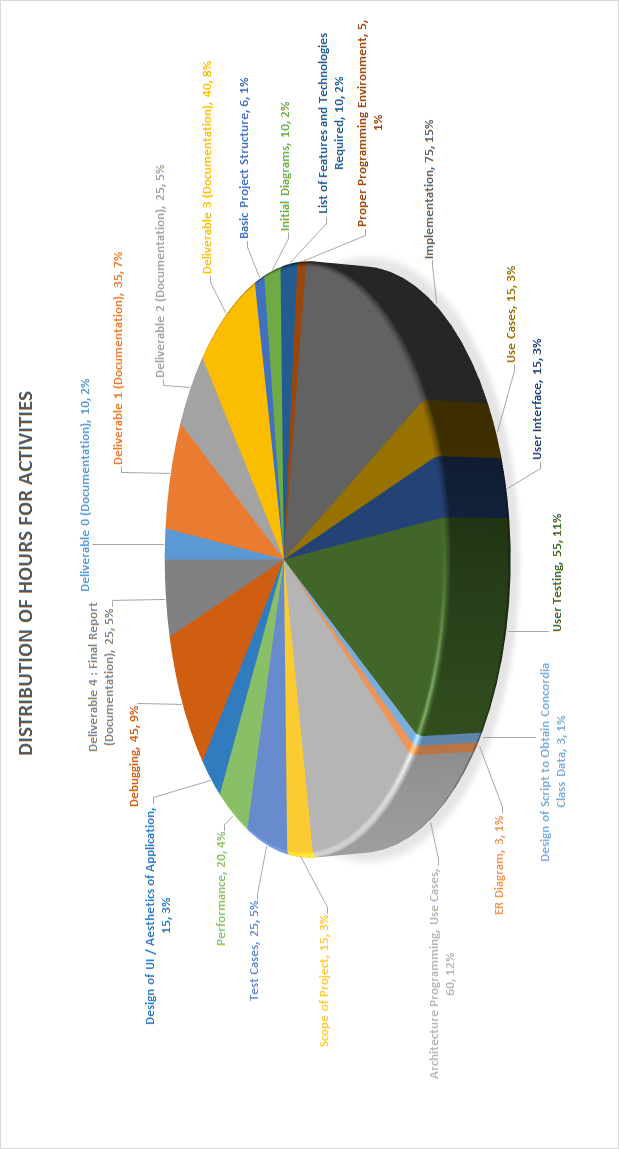


The initial Overall Man-hours for the project after deliverable 2 was 472. The final Overall Man-hours for the project is 512. This is an increase of roughly 8.5%. As a result of the changes mentioned previously, the team has updated the schedule and pie chart to the following:

\*Bars in Blue signify time allocation that has decreased

\*\*Bars in Red signify time allocation that has increased





# 4. Extra Material

Here are another updated version of the Use Case Tables:

UC10 (Save Schedule) is now rendered useless because all schedules are saved automatically by the system, eliminating the need for the user to do this.

UC12 (Delete Schedule) was removed, as it was decided that a student will have to individually remove courses from their schedule if they want to make it blank. This will make it easier for the system to identify future courses that require the currently removed course as a prerequisite so they can be removed as well.

|  |  |
| --- | --- |
| ID: | UC01 |
| Name: | Login |
| Importance (/5): | 5/5 |
| Difficulty (/5): | 2/5 |
| Risk Assessment: | Low |
| Actors: | Student, Administrator |
| Goals: | * To allow students and administrator access to the site’s home page |
| Summary: | By using this feature, the student and administrator will be able to view their profile information and search through schedules to make changes or create new ones. |
| Preconditions: | The user must be on the login page |
| Basic Flow: | 1. The user provides a valid username and password 2. The system responds by checking the username and password 3. The system redirects the user to another page of the website |
| Post Conditions: | Success: The user will be brought to the home page of the website  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, Administrator |
| Goals: | * To allow students and administrators to log off of their account in the system |
| Summary: | Any user can log out of the system to terminate their session |
| Preconditions: | User must have successfully logged into the system |
| Basic Flow: | 1. User selects “Log Out” 2. The System redirects the User to the Sign In 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, Administrator |
| Goals: | * To allow students and administrators 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, password and email and edit it if necessary. |
| Preconditions: | User must have successfully logged into the system |
| Basic Flow: | 1. User selects the “Account” section 2. System redirects user to their “My Account” page 3. System displays all the user’s account information 4. The user edits one of the displayed fields 5. The user selects “Submit Changes” 6. The system redirects the user to the home page |
| Post Conditions: | Success: System redirects User to home page, changes are saved  Failure: User remains on current page, changes unsaved |

|  |  |
| --- | --- |
| ID: | UC04 |
| Name: | Reset Password |
| Importance (/5): | 5/5 |
| Difficulty (/5): | 3/5 |
| Risk Assessment: | High |
| Actors: | Student, Administrator |
| Goals: | * To allow students and administrators 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 selects the “Click here to reset password” 2. System prompts the user for the email address associated to the account 3. User enters the email address 4. System confirms the email address is linked to an account 5. System send email to email address with instructions to change the password |
| Post Conditions: | Success: Email is sent to the user, user remains on current page  Failure: Email is not sent, error message is displayed, user remains on current page |

|  |  |
| --- | --- |
| ID: | UC05 |
| Name: | Add Course |
| Importance (/5): | 5/5 |
| Difficulty (/5): | 3/5 |
| Risk Assessment: | Medium |
| Actors: | Student, Administrator |
| Goals: | * Allow students and administrator to add specified course to a schedule |
| Summary: | Specified courses can be successfully added by the user and to a schedule. |
| Preconditions: | * The user must be logged in and on the home page * The system has access to the course list & schedules * The system has access to previously created schedules * The user must have the pre-requisites (System will check) |
| Basic Flow: | 1. User selects “Modify Schedule” 2. System displays available disciplines to the User 3. User selects the discipline they would like to expand 4. System expands the disciplines to show available courses 5. User selects the course they would like to add 6. User selects “Recompute Schedule” 7. System checks the pre-requisites/co-requisites 8. System prompts the user to select a section 9. User selects a section 10. User selects “Commit Changes” 11. System indicates if the add was successful 12. System displays the course & section in the schedule. |
| 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, Administrator |
| Goals: | * To allow students and administrators to change the section of a course in a schedule |
| Summary: | Specified courses can change their sections in a schedule. |
| Preconditions: | * The user must be logged in and on the home page * The system has access to the course list & schedules * The system has access to previously created schedules * The Course must already be on the schedule |
| Basic Flow: | 1. User selects a specified course on the schedule 2. System displays alternate sections available for the course 3. User selects one of the other available sections of the course 4. User selects “Commit Changes” 5. System changes the course to the desired section, then displays it on 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, Administrator |
| Goals: | * Allow students and administrators to remove courses from a schedule |
| Summary: | Specified courses can be removed by the user from a schedule. |
| Preconditions: | * The user must be logged in and on the home page * The system has access to the course list & schedules * The system has access to previously created schedules * The Course must already be on the schedule |
| Basic Flow: | 1. User selects specified course 2. System displays the details of the course 3. User selects “Remove Course” 4. System removes the course from the schedule, then displays the updated 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 selects “Auto Generate Schedule” 2. System creates schedules and displays them |
| 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 selects “Export Schedule” 2. System begins download of the schedule in PDF format on the computer |
| 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, Administrator~~ |
| ~~Goals:~~ | * ~~To allow students and administrators to save a schedule 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. ~~User selects “save schedule”~~ 2. ~~System prompts the user for confirmation~~ 3. ~~User confirms the saving of the schedule~~ 4. ~~System then saves the schedule and displays it~~ |
| ~~Post Conditions:~~ | ~~Success: Schedule is saved and added to database of the Student~~  ~~Failure: Process fails, Error is displayed to the user~~ |

\*Omitted due to change in scope of Project

|  |  |
| --- | --- |
| ID: | UC11 |
| Name: | View Saved Schedule |
| Importance (/5): | 2/5 |
| Difficulty (/5): | 3/5 |
| Risk Assessment: | Medium |
| Actors: | Student, Administrator |
| Goals: | * To allow students and administrators to view saved schedules |
| Summary: | The user can load any schedule previously created |
| Preconditions: | * The user must have logged in & be on the My Account page * The user must have access to a schedule saved * The system must have access to all saved schedules |
| Basic Flow: | 1. User selects “Account” 2. System redirects user to home page and displays the user’s last updated schedule |
| Post Conditions: | Success: Schedule is displayed from the saved 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 and administrators to remove a saved schedules~~ |
| ~~Summary:~~ | ~~The user can remove any schedule from the list of saved 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 selects “View Saved Schedules”~~ 2. ~~System displays the list of the schedules saved~~ 3. ~~User selects a schedule~~ 4. ~~System displays the schedule~~ 5. ~~User selects “remove”~~ 6. ~~System prompts user for confirmation to remove the schedule selected~~ 7. ~~User confirms the selection~~ 8. ~~System removes the schedule, redirects user to home page~~ |
| ~~Post Conditions:~~ | ~~Success: Schedule is removed from the saved list~~  ~~Failure: Process fails, Error is displayed to the user~~ |

\*Omitted due to change in scope of Project

|  |  |
| --- | --- |
| ID: | UC13 |
| Name: | Modify Section Capacity |
| Importance (/5): | 2/5 |
| Difficulty (/5): | 3/5 |
| Risk Assessment: | Medium |
| Actors: | Administrator |
| Goals: | * Allow administrators to change the student capacity of a section |
| Summary: | The administrator can change the capacity of any section specified. |
| Preconditions: | * The Course and specified Section must exist * The system must have access to all courses and sections |
| Basic Flow: | 1. Administrator searches for a specified course 2. System displays the list of available sections 3. Administrator selects a section 4. System displays the section details 5. Administrator selects “Change Capacity” 6. System prompts Administrator for a new capacity 7. Administrator enters new capacity 8. System changes section capacity and redirects Administrator to the home page |
| Post Conditions: | Success: Section Capacity changed  Failure: Process fails, Error is displayed to the user |

|  |  |
| --- | --- |
| ID: | UC14 |
| Name: | Add Student |
| Importance (/5): | 5/5 |
| Difficulty (/5): | 3/5 |
| Risk Assessment: | Medium |
| Actors: | Administrator |
| Goals: | * Allow administrators to select student in order to make changes to their schedule |
| Summary: | The administrator can change any specified student’s schedule. |
| Preconditions: | * The Student must have an account. * The Student must have valid schedules completed * The System must have access to the Students schedules * The Student must have the pre-requisites for the courses that the Administrator may add |
| Basic Flow: | 1. Administrator enters student name 2. System displays the list of students with the same name 3. Administrator selects the student 4. System displays the schedules made by the student |
| Post Conditions: | Success: Selected schedule displayed to the Administrator, ready for change  Failure: Process fails, Error is displayed to the user |

|  |  |
| --- | --- |
| ID: | UC15 |
| Name: | Remove Student |
| Importance (/5): | 5/5 |
| Difficulty (/5): | 3/5 |
| Risk Assessment: | Medium |
| Actors: | Administrator |
| Goals: | * Once an administrators has modified a student’s schedule, they remove the student from their queue. |
| Summary: | The administrator removes the selected student from the queue after work was completed. |
| Preconditions: | * The Student must have been selected |
| Basic Flow: | 1. Administrator selects “Remove Student” 2. System removes student from the queue and redirects the Administrator to the home page |
| Post Conditions: | Success: Student removed from the queue  Failure: Process fails, Error is displayed to the user |

|  |  |
| --- | --- |
| ID: | UC16 |
| Name: | Request new Account |
| Importance (/5): | 5/5 |
| Difficulty (/5): | 3/5 |
| Risk Assessment: | Low |
| Actors: | Student |
| Goals: | * To allow students the ability to create a new account for the system |
| Summary: | By using this feature, the student will be able to create a new account by providing their first name, last name, password and email to the system. |
| Preconditions: | User does not have an existing account, and is on the login page. |
| Basic Flow: | 1. User selects “Don’t have an account? Create one now” 2. System prompts user to enter account information 3. User enters account information 4. System verifies user email and password combination to be valid and unique 5. System redirects user to main page of website |
| Post Conditions: | Success: System redirects User to home page, account information is saved  Failure: User remains on current page, system displays error |