

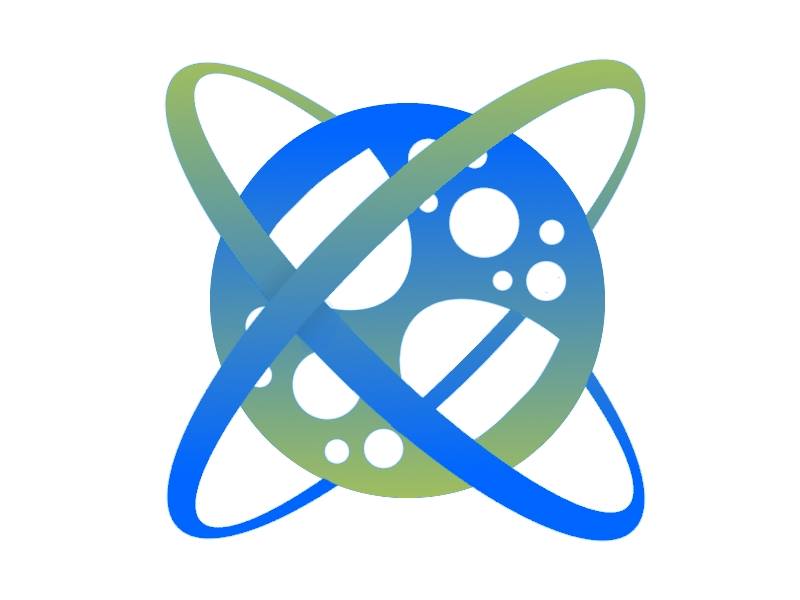
**Software Process, SOEN 341/4 S, Winter 2016**

Dr. Shang

Dr. Fancott

Mr. Morse

**TimeTurner** by team YAWD

****

**Project Testing and Delivery Document – Deliverable 3**

|  |  |
| --- | --- |
| **Team members information** | |
| **Name** | **SID** |
| **Aline Koftikian** | **27764162** |
| **Bryce Drewery-Schoeler** | **27283199** |
| **Claudia Della Serra** | **26766048** |
| **Daniel Di Corpo** | **26331602** |
| **Dimitri Topaloglou** | **29358269** |
| **Erin Benderoff** | **27768478** |
| **Ideawin-Bunthy Koun** | **26314155** |
| **Kevin Yasmine** | **27195346** |
| **Lori Dalkin** | **27738293** |
| **Marc-Andre Leclair** | **27754876** |
| **Philip Lim** | **27485506** |
| **Ryan Lee** | **27752504** |

Grading Sheet

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

DO NOT REMOVE THIS PAGE WHEN SUBMITTING YOUR DOCUMENT

**TABLE OF CONTENTS**

[1. Introduction 6](#_Toc447569629)

[2. Project Description 6](#_Toc447569630)

[3. Testing Report 6](#_Toc447569631)

[3.1 Test Coverage 6](#_Toc447569632)

[3.1.1 Tested Items 6](#_Toc447569633)

[3.1.1.1 Unit Testing: 6](#_Toc447569634)

[3.1.1.2 Requirements Testing: 7](#_Toc447569635)

[3.1.1.3 Stress Testing: 7](#_Toc447569636)

[3.1.1.4 Security Testing: 7](#_Toc447569637)

[3.1.2 Untested Items of Interest 7](#_Toc447569638)

[3.2 Test Cases 8](#_Toc447569639)

[3.2.1 Unit Testing 8](#_Toc447569640)

[3.2.1.1 Code for UserScheduleTest: 8](#_Toc447569641)

[3.2.1.2 Code for LoginForm test: 10](#_Toc447569642)

[3.2.2 Requirements Testing 11](#_Toc447569643)

[3.2.2.1 Login 11](#_Toc447569644)

[3.2.2.2 Logout 12](#_Toc447569645)

[3.2.2.3 Create Course Sequence 12](#_Toc447569646)

[3.2.2.4 Browse Course List 12](#_Toc447569647)

[3.2.2.5 View Course Sequence 12](#_Toc447569648)

[3.2.2.6 Generate Schedule 13](#_Toc447569649)

[3.2.2.7 View Saved Schedules 14](#_Toc447569650)

[3.2.2.8 View Academic Record 14](#_Toc447569651)

[3.2.2.9 Drop Course 15](#_Toc447569652)

[3.2.2.10 Add Course 15](#_Toc447569653)

[3.2.2.11 Saved Generated Schedule 16](#_Toc447569654)

[3.2.2.12 View Weekly Schedule 16](#_Toc447569655)

[3.2.2.13 Manage Courses (Administrator) 17](#_Toc447569656)

[3.2.2.14 Create Course 19](#_Toc447569657)

[3.2.2.15 View Courses (Administrator) 20](#_Toc447569658)

[3.2.2.16 Add User 20](#_Toc447569659)

[3.2.2.17 Search Users 22](#_Toc447569660)

[3.2.2.18 View User 24](#_Toc447569661)

[3.2.2.19 Update User 24](#_Toc447569662)

[3.2.2.20 Delete User 27](#_Toc447569663)

[3.2.2.21 Browse Users 27](#_Toc447569664)

[3.2.3 3.3.3 Stress Testing 28](#_Toc447569665)

[3.2.4 Security Testing 29](#_Toc447569666)

[3.2.4.1 SQL Inject Me LoginForm 29](#_Toc447569667)

[3.2.4.2 SQL Inject Me PreferenceForm 29](#_Toc447569668)

[3.2.4.3 Nikto results: 31](#_Toc447569669)

[4. System Delivery 34](#_Toc447569670)

[4.1 Installation Manual 34](#_Toc447569671)

[4.1.1 Description 34](#_Toc447569672)

[4.1.2 System requirements 34](#_Toc447569673)

[4.1.3 Installation steps 34](#_Toc447569674)

[4.1.4 Virtual Server Installation Testing 41](#_Toc447569675)

[4.2 User Manual 43](#_Toc447569676)

[4.2.1 Student Users 43](#_Toc447569677)

[4.2.1.1 Login 43](#_Toc447569678)

[4.2.1.2 Logout 43](#_Toc447569679)

[4.2.1.3 Browse Course List 44](#_Toc447569680)

[4.2.1.4 Generate Schedule 44](#_Toc447569681)

[4.2.1.5 Save Generated Schedule 48](#_Toc447569682)

[4.2.1.6 View A Saved Schedule 49](#_Toc447569683)

[4.2.1.7 Drop Course 51](#_Toc447569684)

[4.2.1.8 View Academic Record 52](#_Toc447569685)

[4.2.2 Administrator Users 53](#_Toc447569686)

[4.2.2.1 Login 53](#_Toc447569687)

[4.2.2.2 Logout 53](#_Toc447569688)

[4.2.2.3 Manage Courses (Administrator) 53](#_Toc447569689)

[4.2.2.4 Search User 54](#_Toc447569690)

[4.2.2.5 View Course 55](#_Toc447569691)

[4.2.2.6 Edit Course 55](#_Toc447569692)

[4.2.2.7 Delete Course 56](#_Toc447569693)

[4.2.2.8 Create Course 56](#_Toc447569694)

[4.2.2.9 Browse Courses 57](#_Toc447569695)

[4.2.2.10 Add User 58](#_Toc447569696)

[4.2.2.11 Manage Users 60](#_Toc447569697)

[4.2.2.12 Search for Users 62](#_Toc447569698)

[4.2.2.13 Advanced Search for Users 64](#_Toc447569699)

[4.2.2.14 View User 67](#_Toc447569700)

[4.2.2.15 Update User 68](#_Toc447569701)

[4.2.2.16 Delete User 69](#_Toc447569702)

[4.2.2.17 Viewing Users 71](#_Toc447569703)

[5. Final Cost Estimation 73](#_Toc447569704)

[5.1 Final Estimations 73](#_Toc447569705)

[5.1.1 Deliverable 0 Estimation 73](#_Toc447569706)

[5.1.2 Deliverable 1 Estimation 73](#_Toc447569707)

[5.1.3 Deliverable 2 Estimation 74](#_Toc447569708)

[5.1.4 Deliverable 3 Estimation 74](#_Toc447569709)

[5.1.5 Deliverable 4 Estimation 75](#_Toc447569710)

[5.1.6 Final Estimations 75](#_Toc447569711)

[5.2 Updated Gantt Chart 76](#_Toc447569712)

# Introduction

The following document covers the testing involved in the system. A full breakdown of each test case is provided, accompanied by the results. This is useful information in the installation procedure of this software for any users, which is shown alongside the user manual of this section. The types of testing includes unit testing, requirement testing and stress testing. The final topic included is the final cost estimation of the entire project.

# Project Description

The proposed and outlined web application, known as TimeTurner, is designed to auto-generate a student’s course sequence from their first semester up until the end of their degree. It takes into account user input preferences and and any previously completed courses or course prerequisites before creating this sequence. Preferences can be made by the student and include options such as night classes or having particular days off. The application will notify the user if a certain preference suggested results in an impossibility or conflict in the sequence. This sequence generator will be able to create a sequence at any point throughout the user’s degree, if sudden change in circumstances were to arise.

The goal of this application is to simplify the method with which students may decide and schedule their courses. If a course must be redone, the generator can decide what other courses should move where in regards to the remaining courses to be completed, which can be done in seconds, rather than hours. It saves the time of the user, in a simple and efficient manner. Ultimately, the system’s end goal will be to simplify a student’s task of creating their own course schedule in order to allow students to redirect their time to other more important activities, thus making course registration much simpler, quicker, and easier.

# Testing Report

## Test Coverage

Tested Items

Unit Testing:

Unit testing focuses on testing small units of code at a time. It breaks the system down into its individual classes or methods and tests them each separately. Unit testing is important for making sure each piece of an application works as expected, so that these pieces can then be combined with other parts of the application in order to ultimately have a full working system.

Here, Unit testing was done using PHPUnit 5.3. We performed unit tests on two of the model classes in our system, LoginForm and UserSchedule. For each class, a PHPUnit test class was written that included a test for each method in the model class, to ensure that every method works as expected.

Requirements Testing:

Requirements testing breaks a system down based on its requirements, and tests each one of them to ensure they function as expected. It is important for checking which requirements are met and which are not met. Here, each requirements test case includes a test ID, description, expected output, actual output and result. If the actual output matches the expected output, the result is a *pass*. If not, the result is a *fail*. For any failed tests, we can return to the source code used to implement this requirement and identify the bugs that caused it to fail.

Stress Testing:

Stress testing is used to test an application’s performance under extreme conditions of usage, such as a high volume of users or a large data load. It is done in order to determine the system’s robustness and reliability, and helps to identify issues that may only arise under stressful conditions.

There are a number of tools available for stress testing a web application that simulate extreme conditions. Here, we used Apache JMeter, a Java application that works by creating multiple threads that simultaneously access a given web server via HTTP requests. We tested stress conditions for both the login and schedule generation functions of the system.

Security Testing:

Security testing is crucial for any software system to ensure that the system can protect itself against malicious attacks. We used two tools to test security. The first was SQL Inject Me, a Firefox plugin that sends SQL injection strings through the forms of a webpage and checks for any database errors that may occur as a result. The second was Nikto, a web server scanner that uses a Perl script to perform multiple tests against a given server, including checking for dangerous files and checking for outdated versions.

Untested Items of Interest

For the unit tests, we were only able to test 2 of the 11 models in our system. This was due to tight time constraints, as it took a while to set up PHPUnit and write the test code. We chose to test the LoginForm and UserSchedule models because these classes are very important to the system’s functionality; every user needs the LoginForm to access the system, and UserSchedule is needed to generate a schedule. With more time, we would have ideally performed unit tests on the other 9 models in our system (CompletedCourses, ContactForm, Course, PreferenceForm, Prerequisite, Section, Subsection, User and UserSchedules), as well as the controller classes (CourseController, ProfileController, SchedulerController, SiteController and UserController).

For the requirements tests, we tested all requirements except the ones that were scoped out. Scoped out requirements included *create a course sequence*, *weekly schedule view* and *add a course*. With more time, we would have liked to implement these requirements and perform tests on them.

## Test Cases

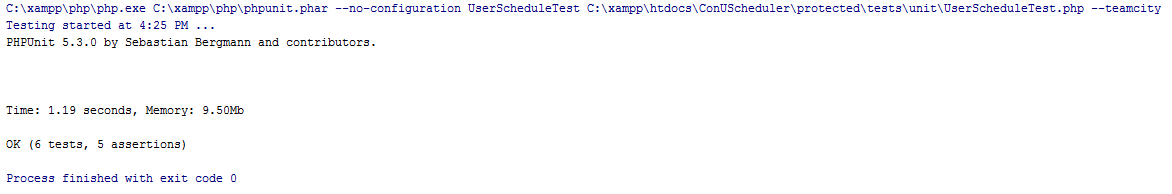
Unit Testing

Code for UserScheduleTest:

**<?php  
require\_once**(**'../../../protected/models/UserSchedule.php'**);  
$yiit=**'../../../framework/yiit.php'**;  
**require\_once**($yiit);  
  
**class** UserScheduleTest **extends** CDbTestCase {  
 **protected $fixture**;  
 **protected $db**;  
  
 **protected function** setUp() {  
 $this->**db** = **new** CDbConnection(**'mysql:host=conuscheduler.ddns.net;dbname=soen341'**,**'soen341'**,**'soen341'**);  
 $this->**db**->**active**=**true**;  
 CActiveRecord::*$db*=$this->**db**;  
 $this->**fixture** = **new** UserSchedule;  
 }  
  
 **protected function** tearDown() {  
 **if**($this->**db**)  
 $this->**db**->**active**=**false**;  
 }  
  
 **public function** testTableName() {  
 $expected = **'user\_schedule'**;  
 $results = $this->**fixture**->tableName();  
 $this->assertEquals($expected,$results);  
 }  
  
 **public function** testRules() {  
 $expected = **array**(  
 **array**(**'scheduleID, courseID, sectionID, subsectionID, year'**, **'required'**),  
 **array**(**'scheduleID, courseID, sectionID, subsectionID, year'**, **'numerical'**, **'integerOnly'**=>**true**),  
 **array**(**'ID, scheduleID, courseID, sectionID, subsectionID, year'**, **'safe'**, **'on'**=>**'search'**),  
 );  
 $results = $this->**fixture**->rules();  
 $this->assertEquals($expected,$results);  
 }  
  
 **public function** testRelations() {  
  
 $expected = **array**(  
 **'scheduleID'** => **array**(UserSchedule::***BELONGS\_TO***, **'UserSchedules'**, **'ID'**)  
 );  
 $results = $this->**fixture**->relations();  
 $this->assertEquals($expected,$results);  
 }  
  
 **public function** testAttributeLabels() {  
 $expected = **array**(  
 **'ID'** => **'ID'**,  
 **'scheduleID'** => **'Schedule'**,  
 **'courseID'** => **'Course'**,  
 **'sectionID'** => **'Section'**,  
 **'subsectionID'** => **'Subsection'**,  
 **'year'** => **'Year'**,  
 );  
 $results = $this->**fixture**->attributeLabels();  
 $this->assertEquals($expected, $results);  
 }  
  
 **public function** testSearch() {  
 $stub = $this->getMockBuilder(**'CDbCriteria'**)  
 ->disableOriginalConstructor()  
 ->getMock();  
 $stub->expects($this->any())  
 ->method(**'compare'**)  
 ->will($this->returnValue(**'true'**));  
 }  
  
 **public function** testModel() {  
 $stub = $this->getMockBuilder(**'UserSchedule'**)  
 ->disableOriginalConstructor()  
 ->getMock();  
 $stub->expects($this->any())  
 ->method(**'model'**)  
 ->will($this->returnValue($this->**fixture**));  
 $this->assertNotNull($stub);  
 }  
}

**Results:**

UserScheduleTest passed 6/6 tests, shown in the figure below:

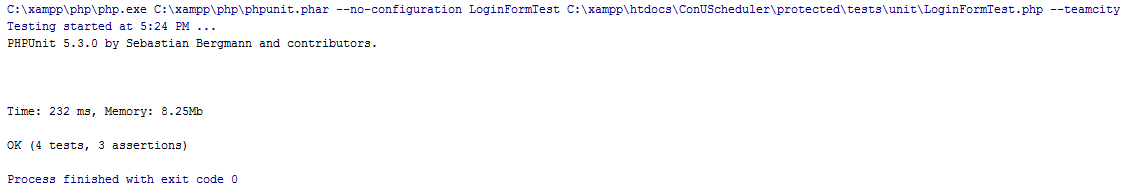


Code for LoginForm test:

**<?php  
  
require\_once**(**'../../../protected/models/LoginForm.php'**);  
**require\_once**(**'../../../protected/components/UserIdentity.php'**);  
$yiit=**'../../../framework/yiit.php'**;  
**require\_once**($yiit);  
  
**class** LoginFormTest **extends** CTestCase  
{  
 **protected $fixture**;  
 **protected function** setUp() {  
 $this->**fixture** = **new** LoginForm;  
 $this->**fixture**->**username** = **"erin"**;  
 $this->**fixture**->**password** = **"soen341"**;  
 $this->**fixture**->**rememberMe** = **"0"**;  
 }  
  
 **protected function** tearDown() {  
 }  
  
 **public function** testRules() {  
 $stub = **array**(**array**(**'username, password'**, **'required'**),  
 **array**(**'rememberMe'**, **'boolean'**),  
 **array**(**'password'**, **'authenticate'**));  
 $results = $this->**fixture**->rules();  
 $this->assertEquals($stub,$results);  
 }  
  
 **public function** testAttributeLabels() {  
 $stub = **array**(**'rememberMe'**=>**'Remember me next time'**,);  
 $results = $this->**fixture**->attributeLabels();  
  
 $this->assertEquals($stub, $results);  
 }  
  
 **public function** testAuthenticate() {  
 $stub = $this->getMockBuilder(**'UserIdentity'**)  
 ->disableOriginalConstructor()  
 ->getMock();  
 $stub->expects($this->any())  
 ->method(**'authenticate'**)  
 ->will($this->returnValue(**'true'**));  
 }  
  
 **public function** testLogin\_2() {  
 $stub = $this->getMockBuilder(**'LoginForm'**)  
 ->disableOriginalConstructor()  
 ->getMock();  
 $stub->expects($this->any())  
 ->method(**'login'**)  
 ->will($this->returnValue(**'true'**));  
 $this->assertEquals(**'true'**, $stub->login());  
 }  
}

**Results:**

LoginFormTest passed 4/4 tests, shown in the figure below:



Requirements Testing

Login

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **UC1** | | **Login (Administrator)** | | | |
| Test ID | Description | | Expected Output | Actual Output | Result |
| 1.1 | Username: Admin  Password: 1234 | | Home Page | Home Page | **Pass** |
| 1.2 | Username: Admine  Password: 1234 | | Incorrect Username  or Password Error | Incorrect Username  or Password Error | **Pass** |
| 1.3 | Username: Admin  Password: 12345 | | Incorrect Username  or Password Error | Incorrect Username  or Password Error | **Pass** |
| 1.4 | Username: Admine  Password: 12345 | | Incorrect Username  or Password Error | Incorrect Username  or Password Error | **Pass** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **UC1** | | **Login (Student)** | | | |
| Test ID | Description | | Expected Output | Actual Output | Result |
| 1.5 | Username: Erin  Password: soen341 | | Home Page | Home Page | **Pass** |
| 1.6 | Username: Erine  Password: soen341 | | Incorrect Username  or Password Error | Incorrect Username  or Password Error | **Pass** |
| 1.7 | Username: Erin  Password: soen341 | | Incorrect Username  or Password Error | Incorrect Username  or Password Error | **Pass** |
| 1.8 | Username: Erine  Password: soen341 | | Incorrect Username  or Password Error | Incorrect Username  or Password Error | **Pass** |

Logout

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **UC2** | | **Logout (Administrator)** | | | |
| Test ID | Description | | Expected Output | Actual Output | Result |
| 2.1 | User clicks logout tab | | Home Page | Home Page | **Pass** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **UC2** | | **Logout (Administrator)** | | | |
| Test ID | Description | | Expected Output | Actual Output | Result |
| 2.2 | User clicks logout tab | | Home Page | Home Page | **Pass** |

Create Course Sequence

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **~~UC3~~** | | **~~Create Course Sequence~~** | | | |
| ~~Test ID~~ | ~~Description~~ | | ~~Expected Output~~ | ~~Actual Output~~ | ~~Result~~ |

Browse Course List

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **UC4** | | **Browse Course List** | | | |
| Test ID | Description | | Expected Output | Actual Output | Result |
| 4.1 | Student requests to view course list | | List of all courses in the system is displayed | List of all courses in the system is displayed | **Pass** |
| 4.2 | Student clicks on a prerequisite course | | Course information for that course is displayed | Course information for that course is displayed | **Pass** |

View Course Sequence

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **~~UC5~~** | | **~~View Course Sequence~~** | | | |
| ~~Test ID~~ | ~~Description~~ | | ~~Expected Output~~ | ~~Actual Output~~ | ~~Result~~ |
| ~~5.1~~ | ~~Student requests to view course sequence with no saved schedule~~ | | ~~Course sequence is displayed~~ |  |  |
| ~~5.2~~ | ~~Student requests to view course sequence after adding a course~~ | | ~~Course sequence is displayed~~ |  |  |
| ~~5.3~~ | ~~Student requests to view course sequence after dropping a course~~ | | ~~Course sequence is displayed~~ |  |  |
| ~~5.4~~ | ~~Student requests to view course sequence after dropping all courses~~ | | ~~Course sequence is displayed~~ |  |  |

Generate Schedule

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **UC6** | | **Generate Schedule** | | | |
| Test ID | Description | | Expected Output | Actual Output | Result |
| 6.1 | Student leaves the preferences blank and generates the schedule for a given year. | | All courses for each semester in each academic year selected is displayed. | All courses for each semester in each academic year selected is displayed. | **Pass** |
| 6.2 | Student selects a time preferences without selecting the day checkbox, then generates schedule for a given year. | | All courses for each semester in each academic year selected is displayed. | All courses for each semester in each academic year selected is displayed. | **Pass** |
| 6.3 | Student selects a day without specifying the time preferences for that day, then generates a schedule for a given year. | | No courses are displayed: ‘no result’ | No courses are displayed: ‘no result’ | **Pass** |
| 6.4 | Student selects a day and the times for that day as a preference, and generates the schedule for a given year. | | Only courses that match the time filter are shown, no result found if no classes match the time criteria. | Only courses that match the time filter are shown, no result found if no classes match the time criteria. | **Pass** |
| 6.5 | Student selects a set of time preferences, but no year and generates the schedule. | | No courses are displayed: ‘no result’. | No courses are displayed: ‘no result’. | **Pass** |
| 6.6 | Student clicks ‘generate schedule button’ with a set of time preferences and for given years, and clicks ‘validate schedule’ without choosing any tutorial and laboratory sections. | | Error message indicating to select tutorial and laboratory section pops up. | Empty error message pops up:  [] | Fail |
| 6.7 | Student choose a day, a set time preferences with the end time before the start time. | | No courses are displayed: ‘no result’ | No courses are displayed: ‘no result’ | **Pass** |

View Saved Schedules

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **UC7** | | **View Saved Schedules** | | | |
| Test ID | Description | | Expected Output | Actual Output | Result |
| 7.1 | Student chooses to view saved schedules | | List of saved schedules is displayed | List of saved schedules is displayed | **Pass** |
| 11.2 | Student chooses to view a specific saved schedule | | Saved schedule is displayed | Saved schedule is displayed | **Pass** |
| 14.3 | Student chooses to delete a saved schedule | | A prompt to confirm deletion is given, then after confirmation the saved schedule is removed | A prompt to confirm deletion is given, then after confirmation the saved schedule is removed | **Pass** |

View Academic Record

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **UC8** | | **View Academic Record** | | | |
| Test ID | Description | | Expected Output | Actual Output | Result |
| 8.1 | Student requests to view Academic Record | | Page with a record of classes passed and the student’s gpa is displayed | Page with a record of classes is displayed, but no classes on the record | **Pass** (with errors when no classes completed) |

Drop Course

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **UC9** | | **Drop Course** | | | |
| Test ID | Description | | Expected Output | Actual Output | Result |
| 9.1 | Student requests to drop a non-prerequisite course, and confirms the choice | | Updated schedule is produced and displayed | Updated schedule is produced and displayed | **Pass** |
| 9.2 | Student requests to drop a non-prerequisite course, and cancels the confirmation dialog box | | Same schedule is displayed | Same schedule is displayed | **Pass** |
| 9.3 | Student requests to drop a course that is a prerequisite to another course saved later in the schedule, and confirms the choice | | Message indicating that that course could not be dropped due to it being a prerequisite for a later course that’s saved is displayed | The course is successfully dropped from the schedule, and the course requiring that prerequisite stays in the schedule | Fail |

Add Course

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **~~UC10~~** | | **~~Add Course~~** | | | |
| ~~Test ID~~ | ~~Description~~ | | ~~Expected Output~~ | ~~Actual Output~~ | ~~Result~~ |
| ~~10.1~~ | ~~Student requests to view list of saved schedules~~ | | ~~List of saved schedules is displayed~~ | ~~List of saved schedules is displayed~~ | **~~Pass~~** |
| ~~10.2~~ | ~~Student requests to view a saved schedule~~ | | ~~A schedule is displayed~~ | ~~A schedule is displayed~~ | **~~Pass~~** |
| ~~10.3~~ | ~~Student requests to add a course to the saved schedule~~ | | ~~A search screen is displayed prompting the user to enter the name of the course~~ | *~~Not Yet Implemented~~* | *~~Not Yet Implemented~~* |

Saved Generated Schedule

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **UC11** | | **Save Generated Schedules** | | | |
| Test ID | Description | | Expected Output | Actual Output | Result |
| 11.1 | Student chooses to save a generated schedule | | Saved schedule confirmation is displayed | Saved schedule confirmation is displayed | **Pass** |
| 11.2 | Student chooses to save a schedule in a semester that already has a saved schedule | | Error message indicating the semester already has a schedule saved | Saved schedule confirmation is displayed | Fail |

View Weekly Schedule

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **~~UC12~~** | | **~~View Weekly Schedule~~** | | | |
| ~~Test ID~~ | ~~Description~~ | | ~~Expected Output~~ | ~~Actual Output~~ | ~~Result~~ |
| ~~12.1~~ | ~~Student requests to see a saved schedule.~~ | | ~~Displays a prompt message to let student select a week.~~ | ~~Not implemented~~ | ~~Not implemented~~ |
| ~~12.2~~ | ~~From the prompted message of week selection, student selects a week.~~ | | ~~Displays the student’s schedule for the selected week.~~ | ~~Not implemented~~ | ~~Not implemented~~ |

Manage Courses (Administrator)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **UC13** | | **Manage Courses** | | | |
| Test ID | Description | | Expected Output | Actual Output | Result |
| 13.1 | Administrator requests to manage courses | | Course management page is displayed | Course management page is displayed | **Pass** |
| 13.2 | Administrator enters an existing course I.D. to the search bar | | Course with the given I.D. is displayed, along with an option to edit the course | Course with the given I.D. is displayed, along with the course code, course description, and an option to view, edit, or delete the course | **Pass** |
| 13.3 | Administrator enters a non-existent course I.D. to the search bar | | Message indicating that a course with that I.D. does not exist | “No results found” message is displayed | **Pass** |
| 13.4 | Administrator enters an existent course code to the search bar | | Course with the given code is displayed, along with an option to edit the course | Course with the given code is displayed, along with the course I.D., course description, and an option to view, edit, or delete the course | **Pass** |
| 13.5 | Administrator enters a non-existent course code to the search bar | | Message indicating that a course with that course code does not exist | “No results found” message is displayed | **Pass** |
| 13.6 | Administrator enters an existent course description to the search bar | | Course with the given description is displayed, along with an option to edit the course | Course with the given description is displayed, along with the course I.D., course code, and an option to view, edit, or delete the course | **Pass** |
| 13.7 | Administrator enters a non-existent course description to the search bar | | Message indicating that a course with that description does not exist | “No results found” message is displayed | **Pass** |
| 13.8 | Administrator requests to delete a course and confirms dialog box | | Updated course bank without the deleted course is produced and displayed | Updated course bank without the deleted course is produced and displayed | **Pass** |
| 13.9 | Administrator requests to delete a course and cancels dialog box | | Same course bank including the chosen course is displayed with no changes | Same screen as before requesting course deletion is displayed | **Pass** |
| 13.10 | Administrator requests to update a course | | Course editing page is displayed | Course editing page is displayed | **Pass** |
| 13.11 | Administrator edits the course code of a course | | Updated course bank with the new code is produced and displayed | Updated course details with the new course code is displayed | **Pass** |
| 13.11 | Administrator edits the course code of a course | | Updated course bank with the new description is produced and displayed | Updated course details with the new course description is displayed | **Pass** |
| 13.12 | Administrator edits the number of credits of a course | | Updated course bank with the new number of credits is produced and displayed | Updated course details is displayed, and the new number of credits is produced | **Pass** |
| 13.13 | Administrator edits the type of the course | | Updated course bank with the new course type is produced and displayed | Updated course details is displayed, and the new course type is produced | **Pass** |

Create Course

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **UC14** | | **Create Course** | | | |
| Test ID | Description | | Expected Output | Actual Output | Result |
| 14.1 | Administrator requests to create a course | | Course creation page is displayed | Create Course page is displayed | **Pass** |
| 14.2 | Administrator enters a course ID, description and amount of credits | | Course info is registered and displayed | Course info is registered and displayed | **Pass** |
| 14.3 | Administrator enters a description and credits, but no course ID | | Error message indicating the course ID field has been left blank | Please fix the following input errors:   * Course Code cannot be blank. | **Pass** |
| 14.4 | Administrator enters a course ID and credits, but no description | | Error message indicating the description field has been left blank | Please fix the following input errors:   * Course Description cannot be blank. | **Pass** |
| 14.5 | Administrator enters a course ID and description, but no credits | | Error message indicating the credits field has been left blank | Please fix the following input errors:   * Credits cannot be blank. | **Pass** |

View Courses (Administrator)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **UC15** | | **View Courses** | | | |
| Test ID | Description | | Expected Output | Actual Output | Result |
| 15.1 | Administrator requests to view courses | | List of all courses in the system is displayed | List of all courses in the system is displayed | **Pass** |
| 15.2 | Administrator clicks the create course link | | Create course page is displayed | Create course page is displayed | **Pass** |
| 15.3 | Administrator clicks the manage course link | | Manage courses page is displayed | Manage courses page is displayed | **Pass** |
| 15.4 | Administrator clicks on a course prerequisite | | Course information for that course is displayed | Course information for that course is displayed | **Pass** |

Add User

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **UC16** | | **Add User** | | | |
| Test ID | Description | | Expected Output | Actual Output | Result |
| 16.1 | Administrator requests to add a user | | Create user page is displayed | Create user page is displayed | **Pass** |
| 16.2 | Administrator clicks the create button | | Error message indicating that all fields have been left blank | Error: please fix the following input errors   * Username cannot be blank * Firstname cannot be blank * Lastname cannot be blank * Net Name cannot be blank * Password cannot be blank | **Pass** |
| 16.3 | Administrator enters a firstname, lastname, netname, password, but no username. | | Error message indicating the Username field has been left blank | Error: please fix the following input errors   * Username cannot be blank | **Pass** |
| 16.4 | Administrator enters a username, lastname, netname, password, but no firstname. | | Error message indicating the Firstname field has been left blank | Error: please fix the following input errors   * Firstname cannot be blank | **Pass** |
| 16.5 | Administrator enters a username, firstname, netname, password, but no lastname. | | Error message indicating the Lastname field has been left blank | Error: please fix the following input errors   * Lastname cannot be blank | **Pass** |
| 16.6 | Administrator enters a username, firstname, lastname, password, but no netname. | | Error message indicating the Net Name field has been left blank | Error: please fix the following input errors   * Net Name cannot be blank | **Pass** |
| 16.7 | Administrator enters a username, firstname, lastname, netname, but no password. | | Error message indicating the Password field has been left blank | Error: please fix the following input errors   * Password cannot be blank | **Pass** |
| 16.8 | Administrator enters a password of less than 6 characters | | Error message indicating the Password is too short | Error: please fix the following input errors   * Password is too short | **Pass** |
| 16.9 | Administrator enters a username, firstname, lastname, netname, and a password. | | User info is registered and displayed | User info is registered and displayed | **Pass** |

Search Users

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **UC17** | | **Search for Users** | | | |
| Test ID | Description | | Expected Output | Actual Output | Result |
| 17.1 | Administrator leaves the search field blank and press ‘Search’ button. | | Displays all users without any filter. | Displays all users without any filter. | **Pass** |
| 17.2 | Administrator enters the ID in the ID search field of an inexistent user. | | No user found. | No user found. | **Pass** |
| 17.3 | Administrator enters the ID in the ID search field of an existent user. | | Displays a row corresponding to the user. | Displays a row corresponding to the user. | **Pass** |
| 17.4 | Administrator uses comparative operators in the ID search field. | | Displays the users according to the comparative operators. | Displays the users according to the comparative operators. | **Pass** |
| 17.5 | Administrator enters a text in the username search field which is not found in any usernames. | | No user found. | No user found. | **Pass** |
| 17.6 | Administrator enters an existing username in the username search field. | | Displays a row corresponding to the user with that username. | Displays a row corresponding to the user with that username. | **Pass** |
| 17.7 | Administrator enters a portion of an existing username in the username search field. | | Displays one or many rows which username contains the text entered. | Displays one or many rows which username contains the text entered. | **Pass** |
| 17.8 | Administrator enters an existent username in the username search field without respecting upper/lower cases of the actual username. | | Displays a row corresponding to the user with that username. | Displays a row corresponding to the user with that username. | **Pass** |
| 17.9 | Administrator enters a text in the first name search field which is not found in any first names. | | No user found. | No user found. | **Pass** |
| 17.10 | Administrator enters an existing first name in the appropriate search field. | | Displays one or many rows which user’s first name contains the text entered as search criteria. | Displays one or many rows which user’s first name contains the text entered as search criteria. | **Pass** |
| 17.11 | Administrator enters a portion of an existing first name in the appropriate search field. | | Displays one or many rows which user’s first name contains the text entered as search criteria. | Displays one or many rows which user’s first name contains the text entered as search criteria. | **Pass** |
| 17.12 | Administrator enters an existent first name in the appropriate search field without respecting upper/lower cases of the actual first name. | | Displays one or many rows which user’s first name contains the text entered as search criteria. | Displays one or many rows which user’s first name contains the text entered as search criteria. | **Pass** |
| 17.13 | Administrator enters a text in the last name search field which is not found in any last names. | | No user found. | No user found. | **Pass** |
| 17.14 | Administrator enters an existing last name in the appropriate search field. | | Displays one or many rows which user’s last name contains the text entered as search criteria. | Displays one or many rows which user’s last name contains the text entered as search criteria. | **Pass** |
| 17.15 | Administrator enters a portion of an existing last name in the appropriate search field. | | Displays one or many rows which user’s last name contains the text entered as search criteria. | Displays one or many rows which user’s last name contains the text entered as search criteria. | **Pass** |
| 17.16 | Administrator enters an existent last name in the appropriate search field without respecting upper/lower cases of the actual first name. | | Displays one or many rows which user’s last name contains the text entered as search criteria. | Displays one or many rows which user’s last name contains the text entered as search criteria. | **Pass** |
| 17.17 | Administrator enters a text in the net name search field which is not found in any last names. | | No user found. | No user found. | **Pass** |
| 17.18 | Administrator enters an existing net name in the appropriate search field. | | Displays one or many rows which user’s net name contains the text entered as search criteria. | Displays one or many rows which user’s net name contains the text entered as search criteria. | **Pass** |
| 17.19 | Administrator enters a portion of an existing net name in the appropriate search field. | | Displays one or many rows which user’s net name contains the text entered as search criteria. | Displays one or many rows which user’s net name contains the text entered as search criteria. | **Pass** |
| 17.20 | Administrator enters an existent net name in the appropriate search field without respecting upper/lower cases of the actual first name. | | Displays one or many rows which user’s net name contains the text entered as search criteria. | Displays one or many rows which user’s net name contains the text entered as search criteria. | **Pass** |

View User

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **UC18** | | **View User** | | | |
| Test ID | Description | | Expected Output | Actual Output | Result |
| 18.1 | Administrator clicks the magnifying glass button from a user’s row. | | Displays the profile of the corresponding user with administrative operations. | Displays the profile of the corresponding user with administrative operations. | **Pass** |

Update User

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **UC19** | | **Update User** | | | |
| Test ID | Description | | Expected Output | Actual Output | Result |
| 19.1 | Administrator clicks the pencil button from a user’s row. | | Displays the editable profile of the corresponding user with current data filled in the editable profile. | Displays the editable profile of the corresponding user with current data filled in the editable profile. | **Pass** |
| 19.2 | From a user’s profile viewed by an administrator, the administrator clicks on the ‘update user’ operation. | | Displays the editable profile of the corresponding user with current data filled in the editable profile. | Displays the editable profile of the corresponding user with current data filled in the editable profile. | **Pass** |
| 19.3 | Administrator attempts to save the updated profile with empty username. | | Displays an error saying that the username cannot be blank. | Displays an error saying that the username cannot be blank. | **Pass** |
| 19.4 | Administrator attempts to save the updated profile with empty first name. | | Displays an error saying that the first name cannot be blank. | Displays an error saying that the first name cannot be blank. | **Pass** |
| 19.5 | Administrator attempts to save the updated profile with empty last name. | | Displays an error saying that the last name cannot be blank. | Displays an error saying that the last name cannot be blank. | **Pass** |
| 19.6 | Administrator attempts to save the updated profile with empty net name. | | Displays an error saying that the net name cannot be blank. | Displays an error saying that the net name cannot be blank. | **Pass** |
| 19.7 | Administrator attempts to save the updated profile with empty password. | | Displays an error saying that the password cannot be blank. | Displays an error saying that the password cannot be blank. | **Pass** |
| 19.8 | Administrator attempts to save the updated profile with an already existing username. | | Displays an error saying that the username is already taken. | Displays an error saying that the username is already taken. | **Pass** |
| 19.9 | Administrator attempts to save the updated profile with an already existing net name. | | Displays an error saying that the net name is already taken. | Displays an error saying that the net name is already taken. | **Pass** |
| 19.10 | Administrator attempts to save the updated profile with a password which length is lower than six characters. | | Displays an error saying that the password is too short and that the minimum length is six characters. | Displays an error saying that the password is too short and that the minimum length is six characters. | **Pass** |
| 19.11 | Administrator attempts to save the updated profile with any combination of invalidity of test cases 19.3 to 19.10 | | Displays a list of all invalidity to be corrected. | Displays a list of all invalidity to be corrected. | **Pass** |
| 19.12 | Administrator hits the save button without changing any fields. | | Updated profile is displayed, all information including password remains unchanged. | Password has been changed | Fail |
| 19.13 | Administrator attempts to save the updated profile with an inexistent username. | | Updated profile is displayed, all remaining information have remained unchanged. | Updated profile is displayed, all remaining information have remained unchanged. | **Pass** |
| 19.14 | Administrator attempts to save the updated profile with a new first name. | | Updated profile is displayed, all remaining information have remained unchanged. | Updated profile is displayed, all remaining information have remained unchanged. | **Pass** |
| 19.15 | Administrator attempts to save the updated profile with a new last name. | | Updated profile is displayed, all remaining information have remained unchanged. | Updated profile is displayed, all remaining information have remained unchanged. | **Pass** |
| 19.16 | Administrator attempts to save the updated profile with an inexistent net name. | | Updated profile is displayed, all remaining information have remained unchanged. | Updated profile is displayed, all remaining information have remained unchanged. | **Pass** |
| 19.17 | Administrator attempts to save the updated profile with a new password which length is greater than six. | | Updated profile is displayed, password for that user has been changed. | Updated profile is displayed, password for that user has been changed. | **Pass** |
| 19.18 | Administrator changes the privilege of the user from student to administrator. | | Updated profile is displayed, user is now administrator. | Updated profile is displayed, user is now administrator. | **Pass** |
| 19.19 | Administrator clicks on the password field but does not change anything, save the profile. | | Update profile is displayed, user information and password have remained unchanged. | Password has been changed. | Fail |

Delete User

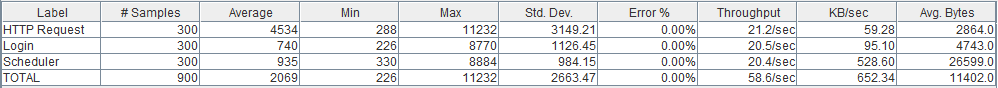
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **UC20** | | **Delete User** | | | |
| Test ID | Description | | Expected Output | Actual Output | Result |
| 20.1 | Administrator hit the ‘X’ button on a row associated to a user in a table of users. | | Displays a confirmation message to delete the corresponding user. | Displays a confirmation message to delete the corresponding user. | **Pass** |
| 20.2 | From the profile page of a user viewed as an administrator, the administrator clicks the ‘Delete User’ operation. | | Displays a confirmation message to delete the corresponding user. | Displays a confirmation message to delete the corresponding user. | **Pass** |
| 20.3 | The administrator hits the cancel button on the confirmation message to delete a user. | | Confirmation message disappears, user remains on the system. | Confirmation message disappears, user remains on the system. | **Pass** |
| 20.4 | The administrator hits the ‘OK’ button on the confirmation message to delete a user. | | The corresponding user is removed from the system, the administrator is brought back to the user management page with an updated table of users. | The corresponding user is removed from the system, the administrator is brought back to the user management page with an updated table of users. | **Pass** |

Browse Users

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **UC21** | | **Browse Users** | | | |
| Test ID | Description | | Expected Output | Actual Output | Result |
| 18.1 | Admin selects user’s tab and is populated with all the users | | List of users are displayed on screen | List of users are displayed on screen | Pass |

3.3.3 Stress Testing

We ran the JMeter tool with 300 simultaneous users and a 5-second ramp-up period (time to get all threads started). There were zero errors, and we obtained the following summary report:

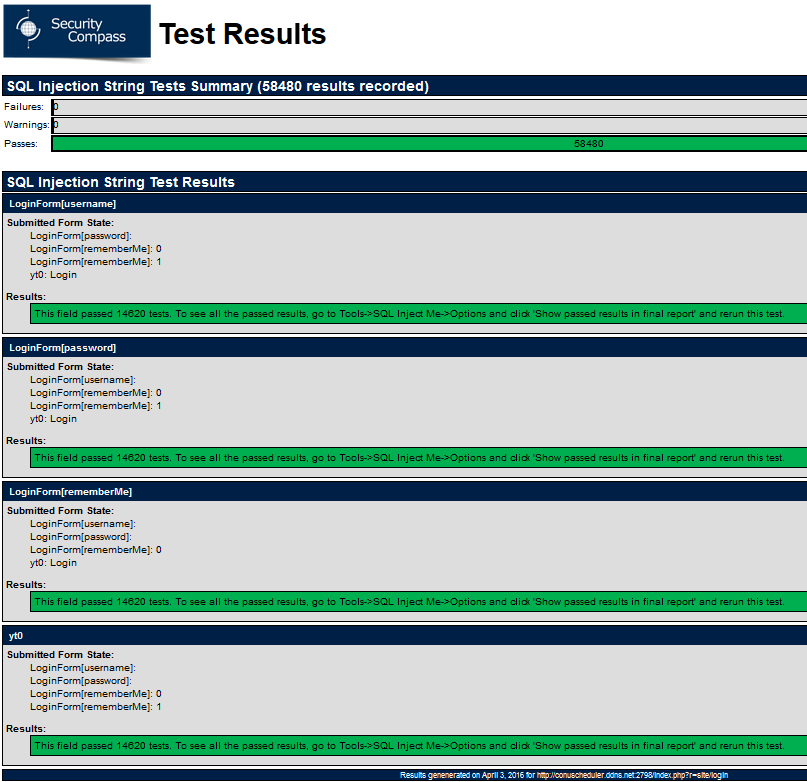


*Figure 3.2.1 JMeter stress testing results*

Security Testing

SQL Inject Me LoginForm

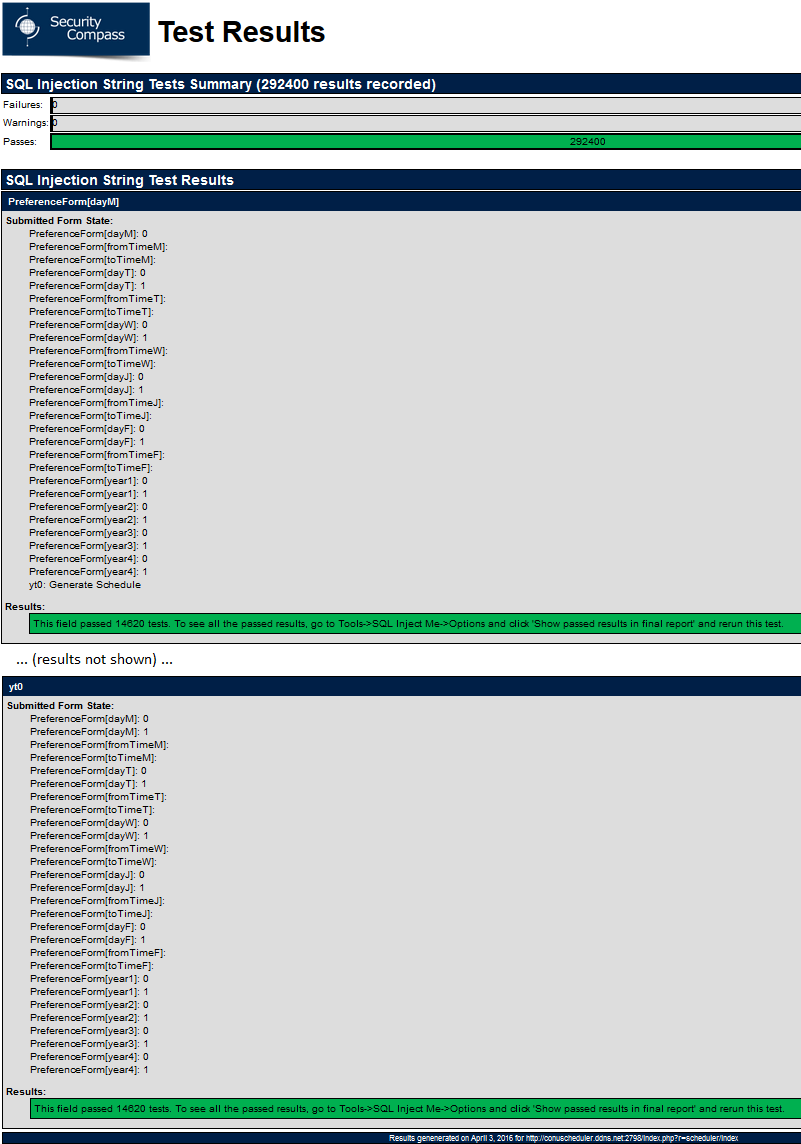
The LoginForm passed all 58480 tests, as depicted in the figure below.

**

*Figure 3.2.2 SQL Inject Me tests on LoginForm*

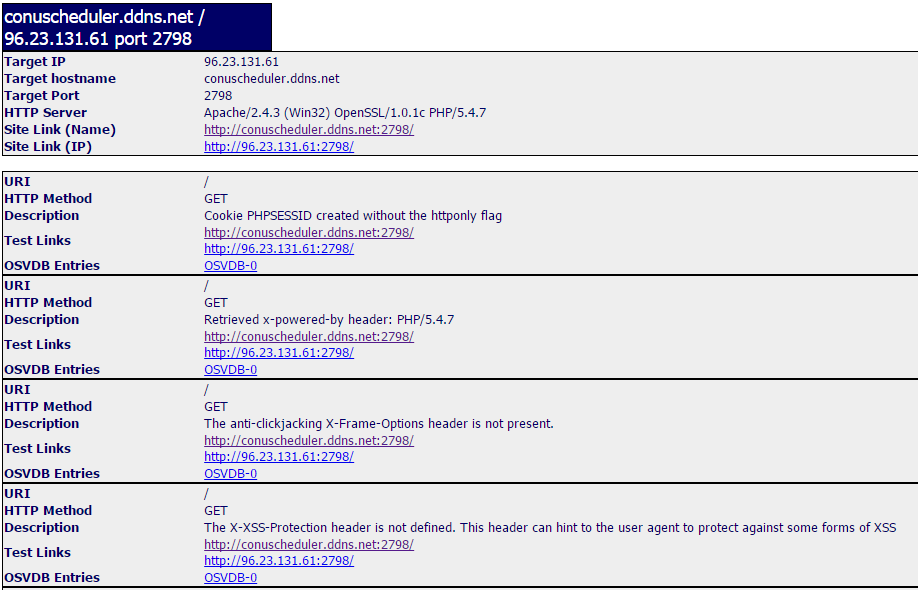
SQL Inject Me PreferenceForm

The PreferenceForm passed all 292400 tests, as depicted in Figure 3.2.3.



*Figure 3.2.3 SQL Inject Me tests on PreferenceForm*

Nikto results:

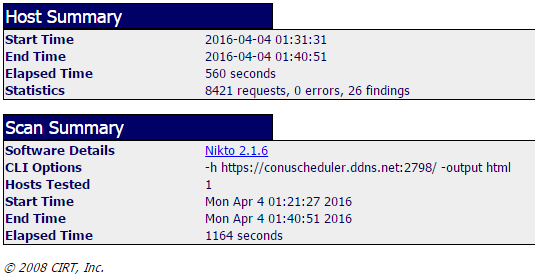












# System Delivery

## Installation Manual

Description

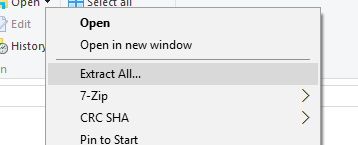
With this product, provided to clients as an archived or zipped folder, only a few simple steps are needed to install this on a client's server. The zip folder includes the root folder of the web application, the database file and Xampp program.

System requirements

* Minimum 2gb of ram
* Windows Vista or newer
* Database included with Xampp,.db file included in zipped folder

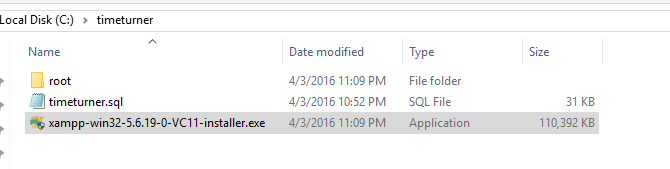
Installation steps

Step 1: Extract the timeturner.zip folder to your preferred location.



*Figure 4.1.1. Folder Extraction*

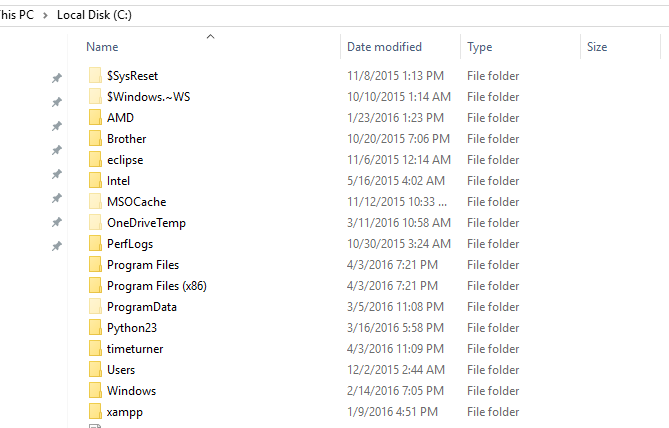
Step 2: Locate the ‘Xampp’ executable and install the program to your preferred location. No special changes made when installing, follow the installation steps



*Figure 4.1.2 Xampp location*

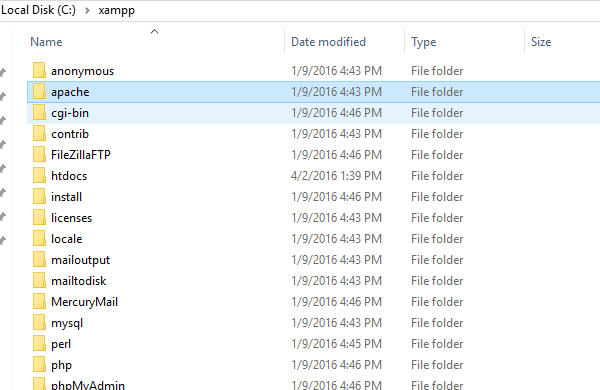
Step 3: Setup the Virtual Server

Step 3.1: Go to the installed ‘Xampp’ folder



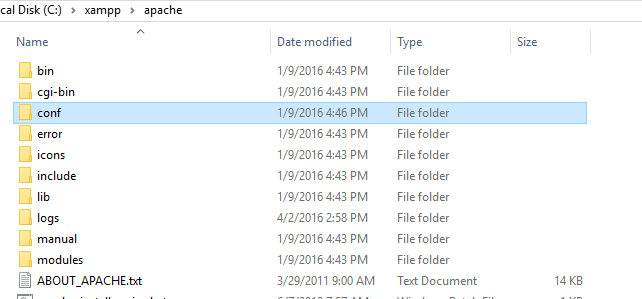
*Figure 4.1.3. Xampp folder location (last folder)*

Step 3.2: Go to the Apache folder



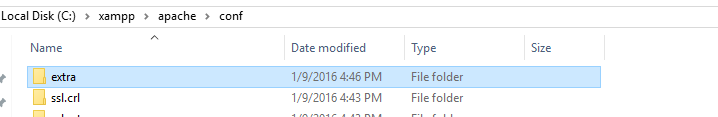
*Figure 4.1.4 Apache folder location*

Step 3.3: Locate conf folder in apache



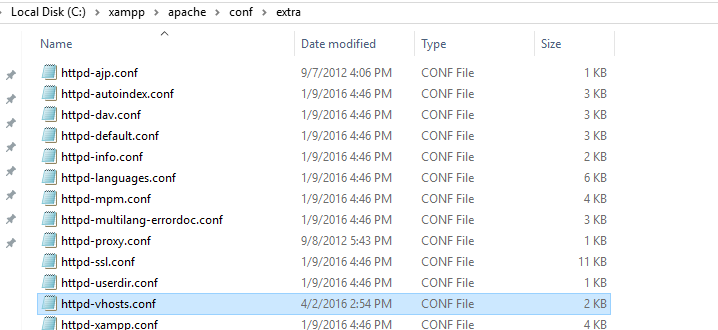
*Figure 4.1.5 conf folder location*

Step 3.4: Go to extra



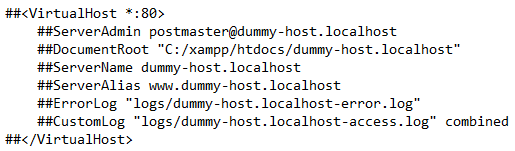
*Figure 4.1.6. Extra folder location*

Step 3.5: Open the httpd-vhost.conf file



*Figure 4.1.7 httpd-vhost.conf file*

Step 3.6: Set up virtual server by including the appropriate lines of code provided and visualized in the next screen capture:



*Figure 4.1.8 Virtual server setup code*

Remove the hashtags from all the lines. The ServerName should be the name of the domain that will be used to access the application via URL. The ServerAlias allows to access the domain through a ‘www’ protocol. For example, if the preferred domain is ‘[www.mytimeturner.com](http://www.mytimeturner.com)’ the ServerName would be **mytimeturner** and the ServerAlias would be ‘[www.mytimeturner.com](http://www.mytimeturner.com)’. The DocumentRoot is the location of the web application root folder, which is provided in the zip folder of the application. The **root** folder provided in the zip file, should be extracted in the same location as the DocumentRoot location. Once provided with valid DocumentRoot location folder, extract the ‘root’ folder in that location.

Step 4: Redirecting windows hosts

Step 4.1: Once the virtual host is setup, the new domain name that was provided in the ServerAlias must be also setup in the Windows host file. To locate the Windows host file:

C:\Windows\System32\Drivers\etc\host.file

Step 4.2: Open the file and append the following lines:

127.0.0.1 [www.mytimeturner.com](http://www.mytimeturner.com)

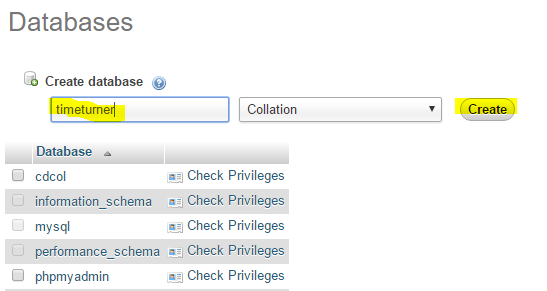
*This simply tells the operating system to resolve the domain as the local machine. Of course, [www.mytimeturner.com](http://www.mytimeturner.com) should match exactly what was setup in the Virtual Host setup ServerAlias name. Save the file.*

**NOTE: Do NOT continue installation after this step until conducting installation testing (see Section 4.1.2).**

Step 5: Database Installation

Step 5.1: To install the database, a database file is provided in the ZIP folder called ‘timeturner.db.’ This file is to be extracted in phpmyadmin.

Step 5.2: At this point, we need to create a database in order to import the existing database file. Click on database, and create a database under the name: ‘timeturner’

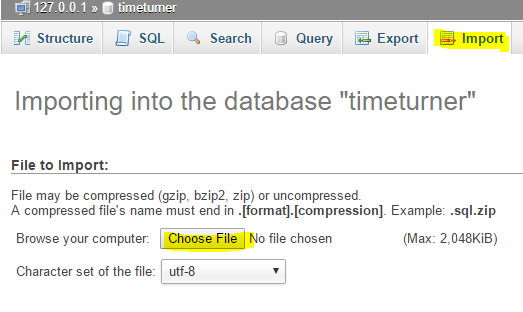


*Figure 4.1.9 Create database*

Step 5.3: Click on ‘Create’. This will create a new database that will appear on the left hand column where the rest of the databases are located. From here, the actual database needs to be imported in order for the application to run.

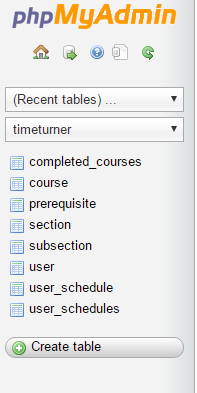
Step 6: Database Extraction

Step 6.1: To extract the database, click on the database name ‘timeturner’ on the left and then click on import.



*Figure 4.1.10 Database extraction*

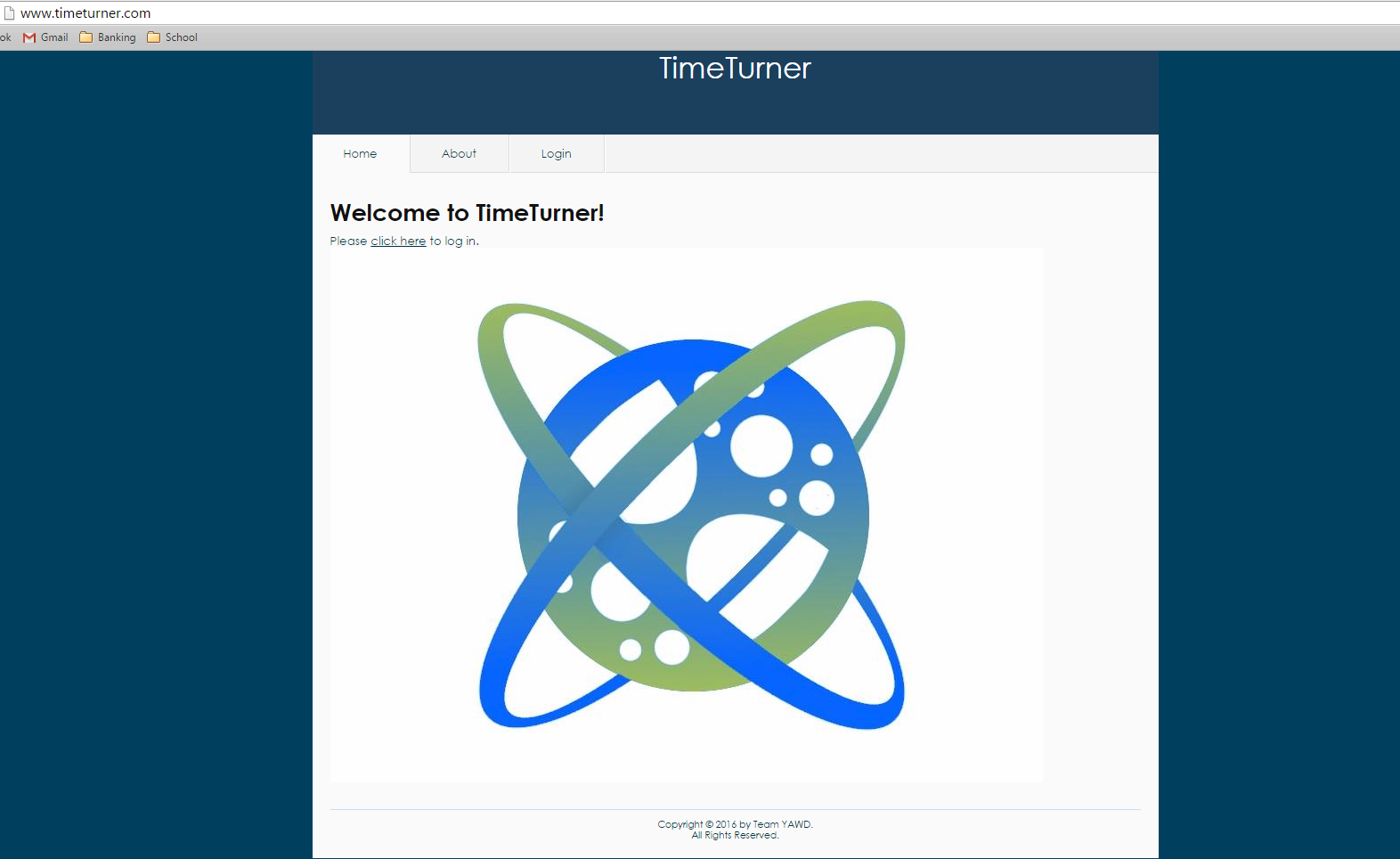
Step 6.2: Upload the ‘timeturner.sql’ file provided and then click on ‘Go’. At this point, all the data tables should appear in the timeturner database:



*Figure 4.1.11 Database tables*

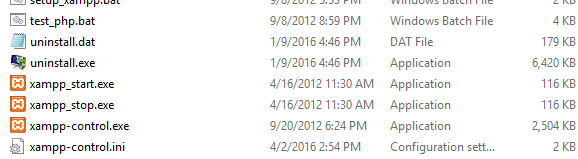
Final Installation Testing

To run the application, open a web browser and type in the domain name provided earlier through ServerAlias, which in this case would be ‘[www.timeturner.com](http://www.timeturner.com)’



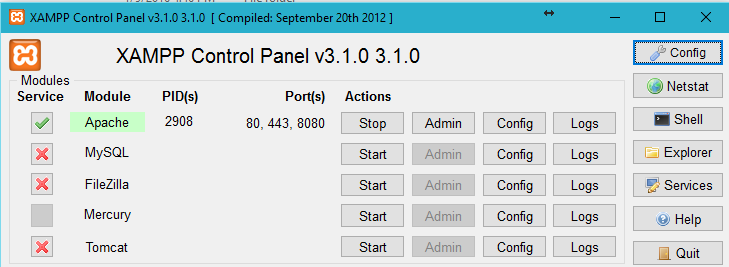
Virtual Server Installation Testing

In order to test if the virtual server was setup properly, locate the ‘XAMPP’ installation folder and locate the xampp-control.exe file:



*Figure 4.1.12 xampp-control.exe file*

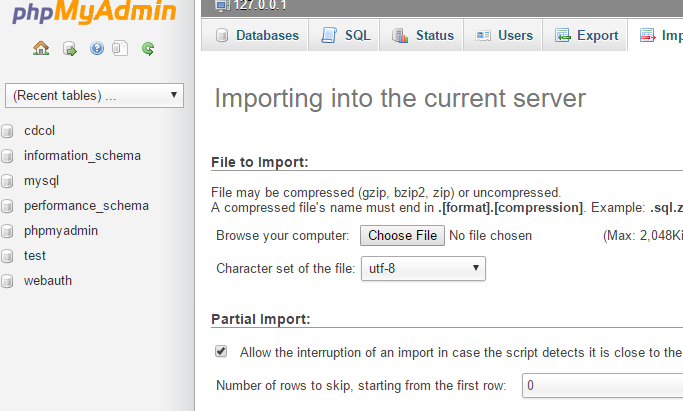
Once XAMPP runs, click on ‘Start’ on the Apache line. If the server was setup properly, the following should appear:



*Figure 4.1.13 Proper running of Xampp server*

A green indicator should appear on Apache.

Once the Apache server is setup, click on ‘Start’ for the MySQL line. Again, a green indicator light should highlight the MySQL. If the green light appears, click on ‘Admin’. This will open up phpmyadmin, a GUI provided by XAMPP for the MySQL database.



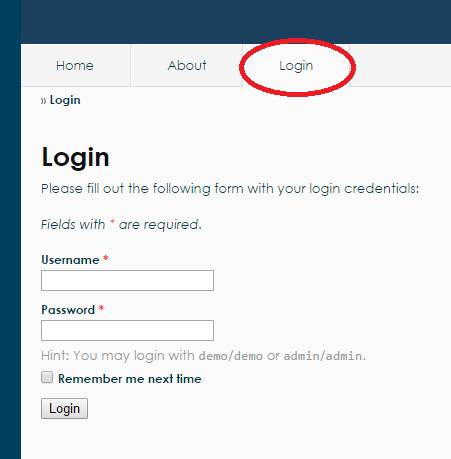
*Figure 4.1.14 phpmyadmin GUI*

## User Manual

Student Users

Login

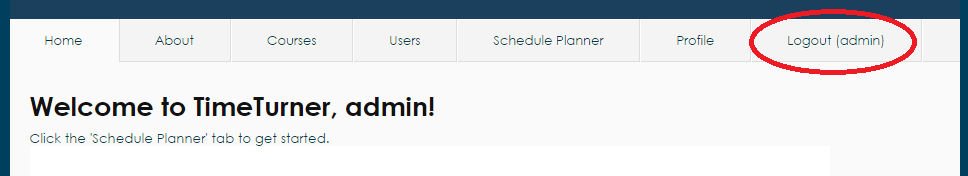
To log in, head to the ‘Login’ tab from the main screen as outlined below and fill out the user information (see Figure . Finally, press the ‘Login’ button at the bottom, and if you wish for the browser to remember the identification, select the “Remember me next time” checkbox.



*Figure 4.2.1. Login tab when not logged into the system*

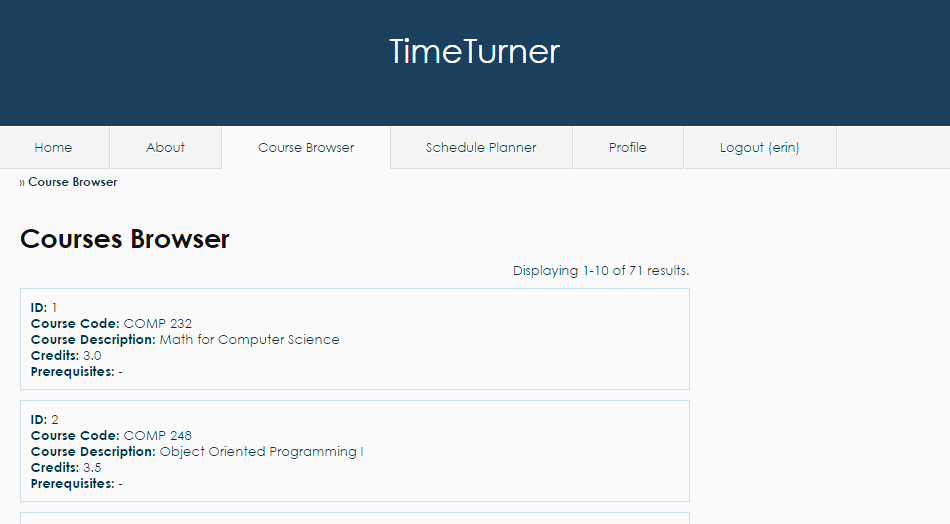
Logout

To log out, simply press the ‘Logout’ button as outlined below, and you will be redirected to the main page.



*Figure 4.2.2 Logout Tab when logged into the system*

Browse Course List



*Figure 4.2.2. Browse Course List functionality*

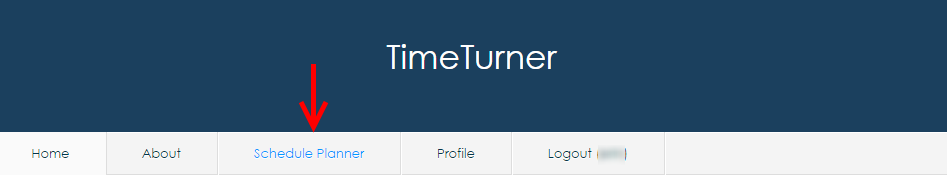
If the user wishes to view the list of all available courses, the user must click on the ‘Course Browser’ tab, and a list will be displayed in an interval of 10 courses (see Figure 4.2.3). To view more courses, the user may scroll down the page and click on the desired page or click ‘Next’ to view the next 10 courses (see Figure 4.2.4).



*Figure 4.2.3. View more courses*

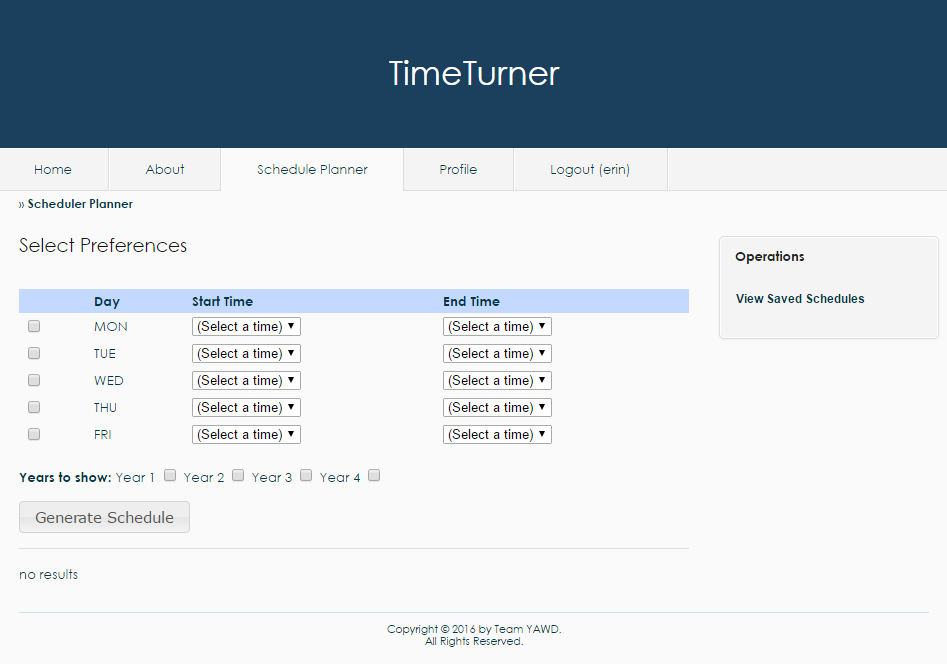
Generate Schedule

Students can generate a schedule based on preferences. To do so, students may go to the ‘Schedule Planner’tab from the home page after logging in (see Figure 4.2.5).



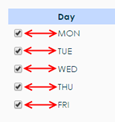
*Figure 4.2.5 Accessibility to the ‘Schedule Planner’ tab from the home page.*

Once the tab is selected, a set of preferences is displayed along with the ‘Generate Schedule’ button and a set of operations (see Figure 4.2.6).



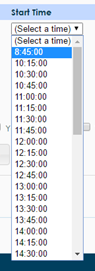
*Figure 4.2.6. Set of preferences displayed on the ‘Schedule Planner’ tab.*

The preferences include the starting and ending time at which the student would prefer having classes for each week day. Students can choose to apply the preference filters for the days desired by selecting the checkboxes corresponding to the day (see Figure 4.2.7).



*Figure 4.2.7. Checkboxes corresponding to the day to be filtered.*

Students may then select the times at which they wish to start and end their courses by scrolling down and choosing the offered times from the drop-down menus (see Figure 4.2.8).



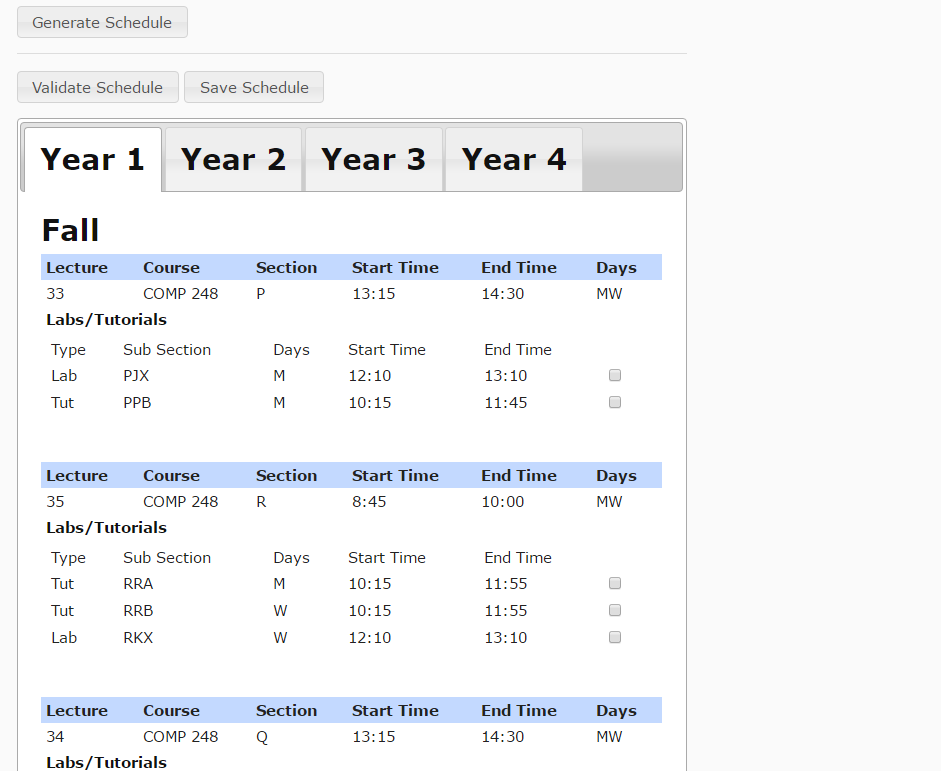
*Figure 4.2.8. Drop-down menu to select time preferences.*

Once the preferences are set, the year(s) to be displayed can be chosen by click the appropriate checkboxes (see Figure 4.2.9). The system will generate schedule based on the preferences only for the years selected.



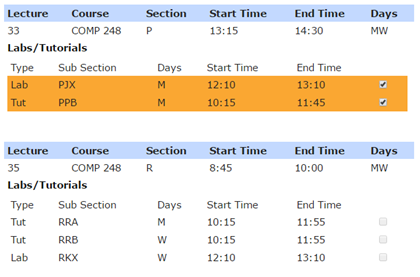
*Figure 4.2.9. Academic years to be displayed according to the set preferences.*

The next step is then to generate the schedule by clicking the ‘Generate Schedule’button. A set of tabs, each representing an academic year, will be displayed on the bottom of the page. Each tab is divided into semesters. Every semester division contains a list of proposed lectures, tutorials and laboratories for all courses in the sequence that correspond to the time blocks selected (see Figure 4.2.10).



*Figure 4.2.11. Tabs representing each academic year selected with courses for each semester.*

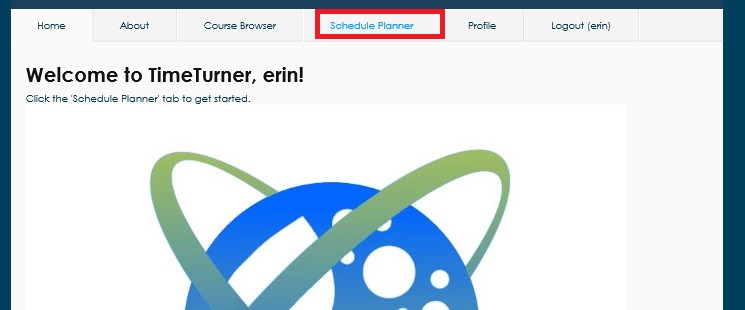
All there is left to do is to choose the laboratory and tutorial times for each of the courses by selecting the checkbox corresponding to the desired sections (see Figure 4.2.12). Once sections are chosen, the student may click the ‘Validate Schedule’button to ensure that the times are consistent and that no overlapping occurs. *.* Note that the checkboxes for tutorials and laboratories of other lecture sections of the same course have been disabled when one lecture section has been selected.



*Figure 4.2.12. Selection of the laboratory and tutorial sections*

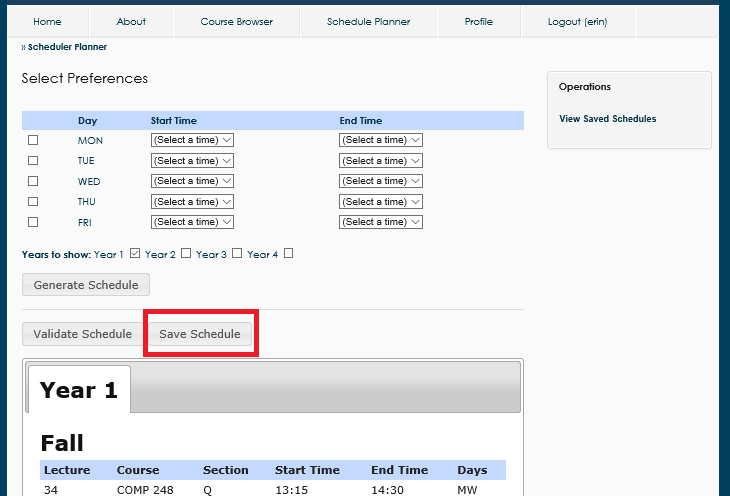
Save Generated Schedule

While on Schedule Planner tab (located at the top of the page (See figure 4.2.13)), the Student may wish to save a schedule they have generated.



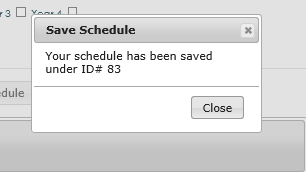
*Figure 4.2.13. Schedule Planner tab*

A schedule that adheres to the specifications requested in the preferences list must be generated firsts. After having generating the schedule for a specific year and selecting the section of choice, the schedule can be saved with the save schedule button (see Figure 4.2.14).



*Figure 4.2.14. Save Schedule button after generating a schedule and choosing tutorial and lab sections*

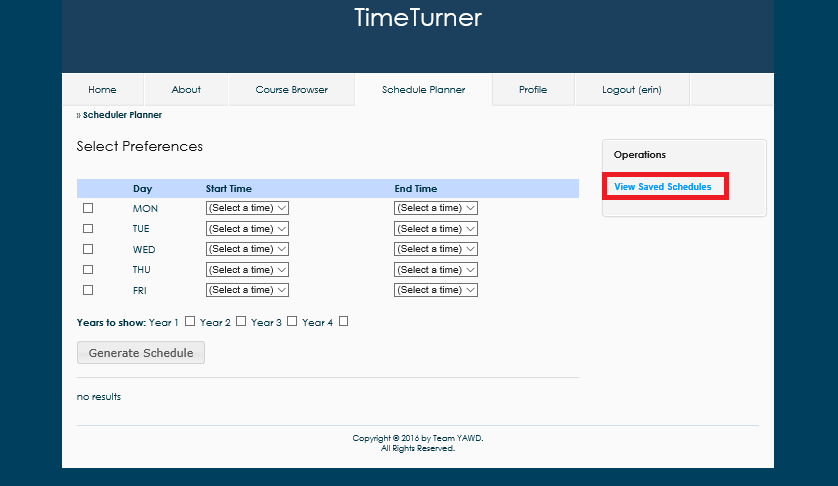
This will return a pop-up indicating confirmation that the schedule has been saved (see Figure 4.2.15).



*Figure 4.2.15. Confirmation of a Saved Schedule*

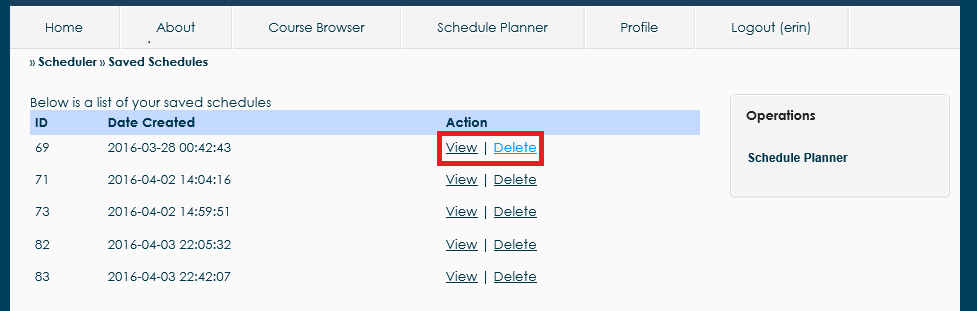
View A Saved Schedule

Upon accessing the Scheduler Planner tab, generated a schedule and saved a schedule, the student may wish to view any schedules they have saved. In the operations box on the sidebar, there is a view saved schedules option (see Figure 4.2.16).



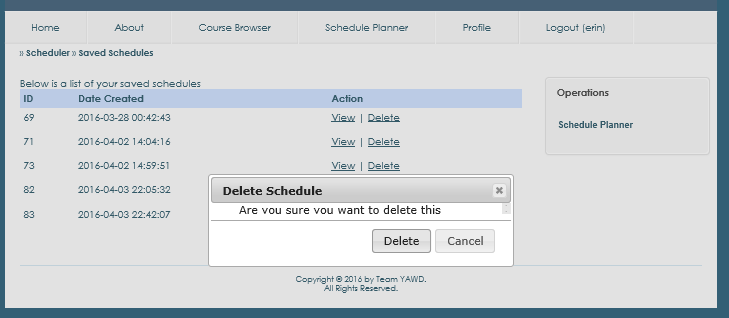
*Figure 4.2.16. View Saved Schedule option*

Upon clicking this option, a list of all saved schedules will be presented to the student. At this point they can view or delete a schedule (see Figure 4.2.17).



*Figure 4.2.17. Options to view or delete a schedule*

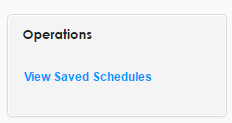
By clicking view, it returns the schedule for the given semester. By deleting it, the system asks for confirmation of deletion and then removes the schedule from the list of saved schedules (see Figure 4.2.18).



*Figure 4.2.18. Delete a schedule confirmation box.*

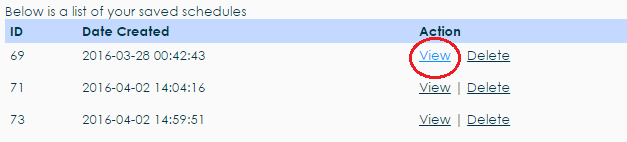
Drop Course

A student that is logged in the system and has at least one saved schedule containing at least one course has the option to drop a course. They can do so by clicking on the “View Saved Schedules” option in the Schedule Planner tab. The option appears on the right hand side of the Schedule Planner page (see Figure 4.2.19).



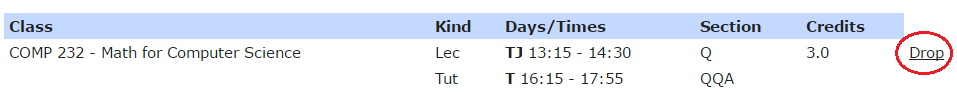
*Figure 4.2.19. View Saved Schedules operation*

A page displaying all currently saved schedules for the student is displayed. To choose the schedule from which a course will be dropped, the “View” option should be selected. It appears to the right of each saved schedule (See Figure 4.2.20).

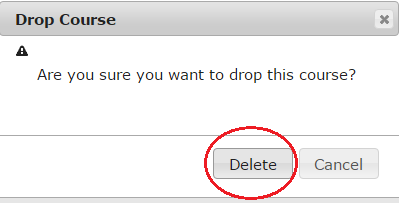


*Figure 4.2.20. View Saved Schedule action*

The selected saved schedule will load, displaying all courses that are currently in that schedule. Each course has a “Drop” option to the right of the number of credits (see Figure 4.2.21).

*Figure 4.2.21. Drop Course action*

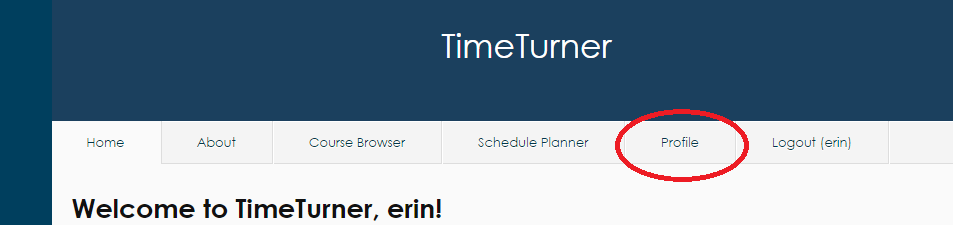
A dialog box prompting the user to confirm their choice will appear after clicking the Drop button. To successfully drop the selected course, the “Delete” button on the left will confirm the deletion of that course from the selected saved schedule (see Figure 4.2.22). If the student changes their mind, the cancel button will cancel the operation and the course is not dropped from the schedule.



*Figure 4.2.22. Confirmation dialog box*

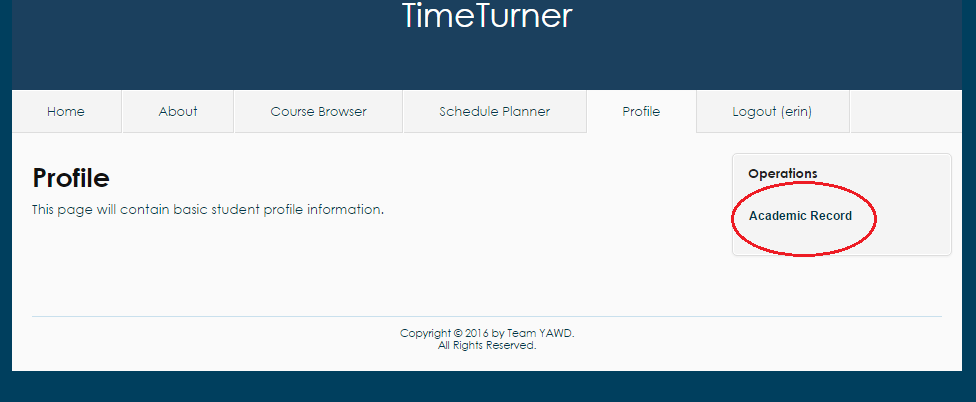
View Academic Record

A student wishing to view their academic record and/or GPA may do so from the ‘Profile’ tab (See Figure 4.2.23).

**

*Figure 4.2.23. Profile Tab located second from right at the top of the screen*

Once on the ‘Profile’ page, the student may click on the ‘View Academic Record’ displayed at the far right of the page (see Figure 4.2.24). Clicking this link will result in the display of all classes that the student has completed, as well as their grades for each course, and their GPA.

**

*Figure 4.2.24. Academic Record link in ‘Operations’ on far right of ‘Profile’ screen*

Administrator Users

Login

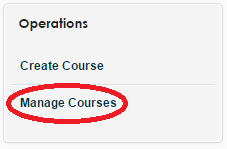
Administrators are simply privileged users. As such, logging in as an Administrator is performed much as logging in as a student. See section 4.2.1.1.

Logout

Administrators are simply privileged users. As such, logging in as an Administrator is performed much as logging in as a student. See section 4.2.1.2.

Manage Courses (Administrator)

An administrator that is logged in the system will have the option to manage courses. This option is found in the Courses tab, on the right hand side of the page (see Figure 4.2.25).

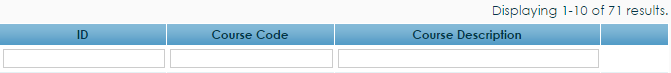


*Figure 4.2.25. Manage Courses operation*

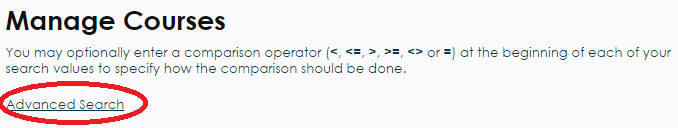
Clicking the manage courses option will display the manage courses page. There are many possible actions for the administrator to take from here.

Search User

If the Administrator wishes to manage a specific course, the search bars above the table containing all the courses may be used to search for a course (see Figure 4.2.26), or the “Advanced Search” setting may be used (see Figure 4.2.27).

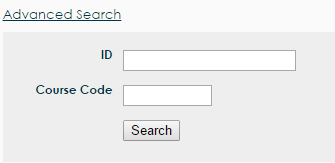


*Figure 4.2.24. Course search bars*



*Figure 4.2.27. Advanced search setting*

The advanced search setting will display two search bars on the page which will accept a course I.D. and/or course code to search the system (see Figure 4.2.28). Clicking the search button will perform the search.



*Figure 4.2.28. Advanced search bars*

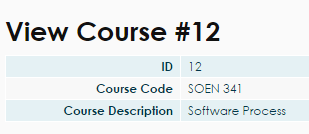
Once a course is searched and found, the administrator has the options of viewing, editing or deleting the course. These options can be seen on Figure 4.2.29, in the respective order.



*Figure 4.2.29. View, edit, and delete buttons*

View Course

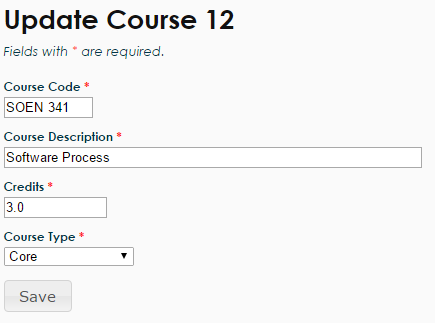
The view option will produce a page that displays the details of the course, including course I.D., course code, and course description (see Figure 4.2.30)



*Figure 4.2.30. View course table*

Edit Course

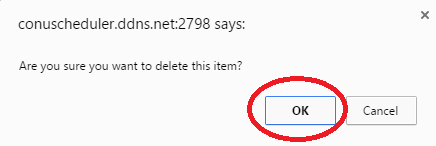
The edit function is selected by clicking the pencil icon to the right of the view button. Clicking the edit button will produce the update course page, which can edit the course code, description, number of credits, and type of the course (see Figure 4.2.31). Any changes made will be finalized by clicking the “Save” button below the editing boxes.



*Figure 4.2.31. Update course boxes*

Delete Course

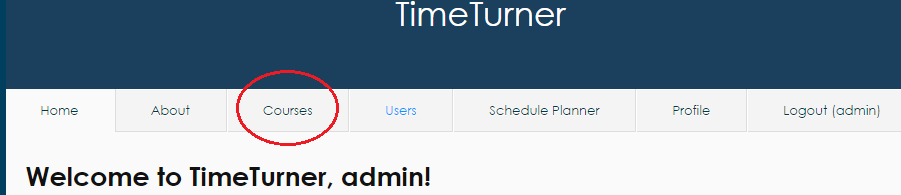
The final operation that an administrator can do to manage a course is to delete a course, which can be performed by clicking the red “X” button to the right of the update course icon. Clicking the X icon will produce a dialog box that prompts the administrator to confirm their decision. To confirm their choice, the “OK” button should be pressed and the course will be deleted from the course bank (see Figure 4.2.32). If the administrator does not want to delete the course, they can simply press the cancel button or exit the dialog box.



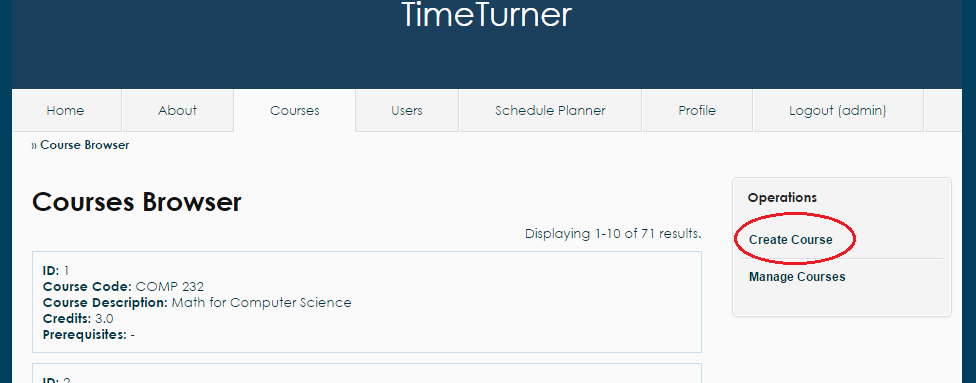
*Figure 4.2.32. Confirmation dialog box for course deletion*

Create Course

An administrator may wish to create a course to add to the database of courses that students can take. In order to do so, the administrator must first click on the ‘Courses’ tab to find themselves on the courses database page (see Figure 4.2.33). From here, the administrator must click the ‘Create Course’ link located on the far right of the page; this action will display the course creation page (see Figure 4.2.34).

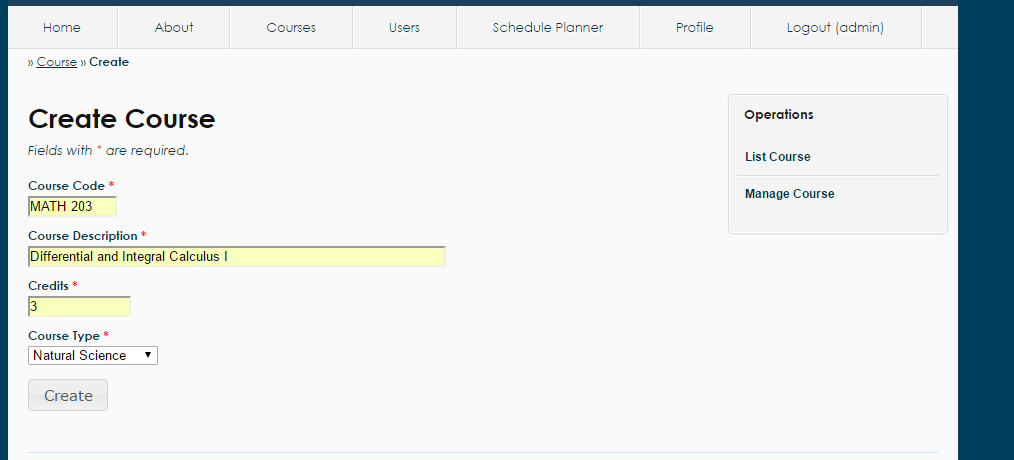
**

*Figure 4.2.33. Courses tab on top of page when logged in as Administrator*

**

*Figure 4.2.34. Create course link on far right of Course Browser page*

This page will ask for the course code, the course description, and the amount of credits awarded by the course (see Figure 4.2.35 for example input). Once the administrator enters the appropriate course information, the system will log such information in the database, and the course information for the newly created course will be displayed on screen, along with the course’s newly assigned ID.

**

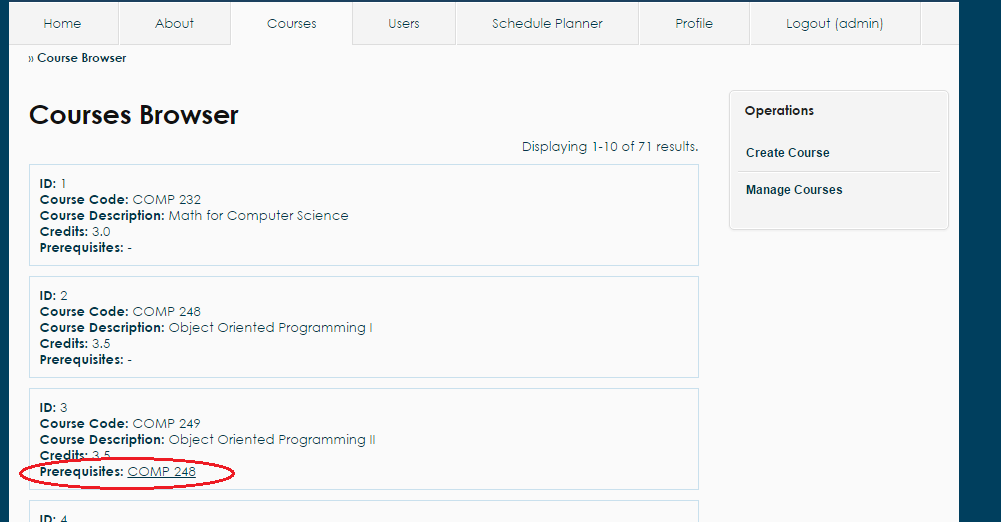
*Figure 4.3.35. Sample input for Create Course functionality*

Browse Courses

While logged in, the administrator may wish to view the database of courses in the system. To accomplish this, the administrator may simply click on the Courses tab; this will cause the list of all courses to be displayed on screen (see Figure 4.2.33). Courses are grouped by page, 10 to a page. If the administrator wishes to view more courses, he may scroll down to the bottom of the page and click on the desired page number.

While on the Courses tab, the administrator may navigate away by clicking the two links located in the ‘Operations’ box to the right of the page: Create Course and Manage Course. Clicking these links will bring the administrator to those respective pages where other activities and actions may be carried out.

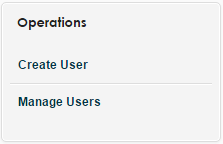
If the administrator wishes to view the prerequisite information of a course, he may simply click on the name of the prerequisite to that course (see Figure 4.2.36). Clicking on the name of the prerequisite will display the course information of the prerequisite.

**

*Figure 4.2.36. Prerequisites for each Course located in the course information box*

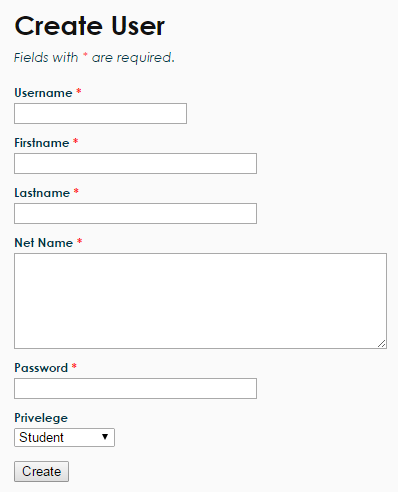
Add User

Once logged in, the administrator can add a new user into the system by first clicking on the ‘Users’ tab.



*Figure 4.2.37. Create user operation*

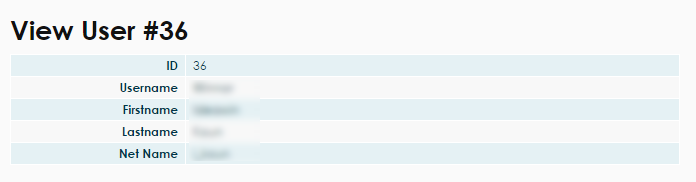
A new page will be loaded, displaying a list of users. On the right side menu, clicking on the ‘Create User’ link (see Figure 4.2.37) will redirect the administrator to another page where information about the new user can be entered. (see Figure 4.2.38).



*Figure 4.2.38. Create user form*

All fields of the form must be completed to be valid. The password must be at least 6 characters long and this will be the password used to log into the system for the user, along with the username. The privilege options allow the administrator to select whether the new user is a student or an administrator.

Upon completing all fields of the form, the administrator may click on the ‘Create’ button at the bottom to submit it. If the user was successfully created, the newly created user’s information will be displayed on another page to the administrator, along with an ID number (see Figure 4.2.39).

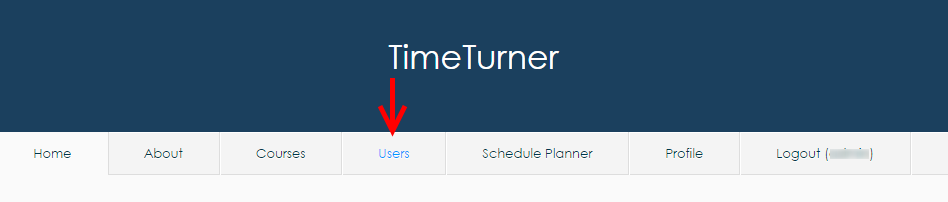


*Figure 4.2.39. View the newly created user’s information*

Manage Users

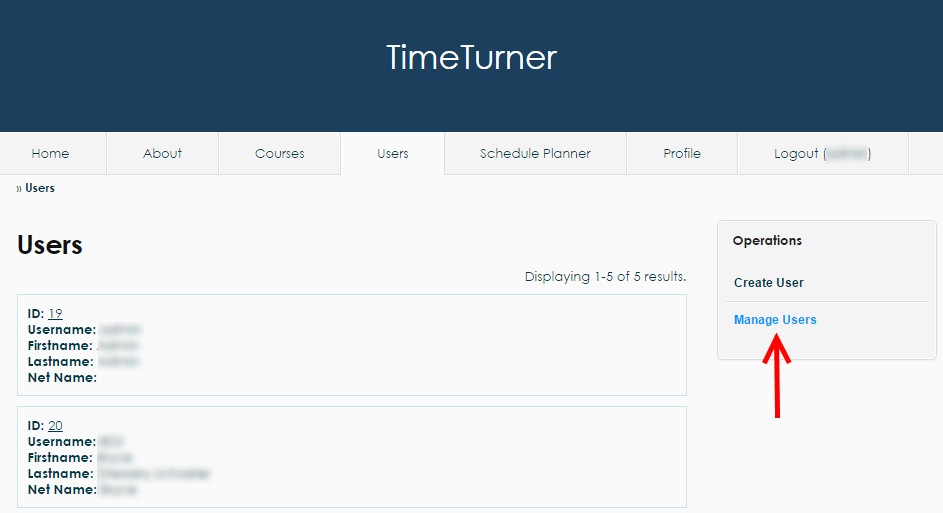
When logged in as an administrator, users of the system can be managed. More precisely, search can be conducted on the system in order to find one or more specific users; information about those users can be viewed and updated, and users can be deleted through this managing section.

In order to manage users, the administrator must first go to the ‘Users’tab from the home page (see Figure 4.2.40).



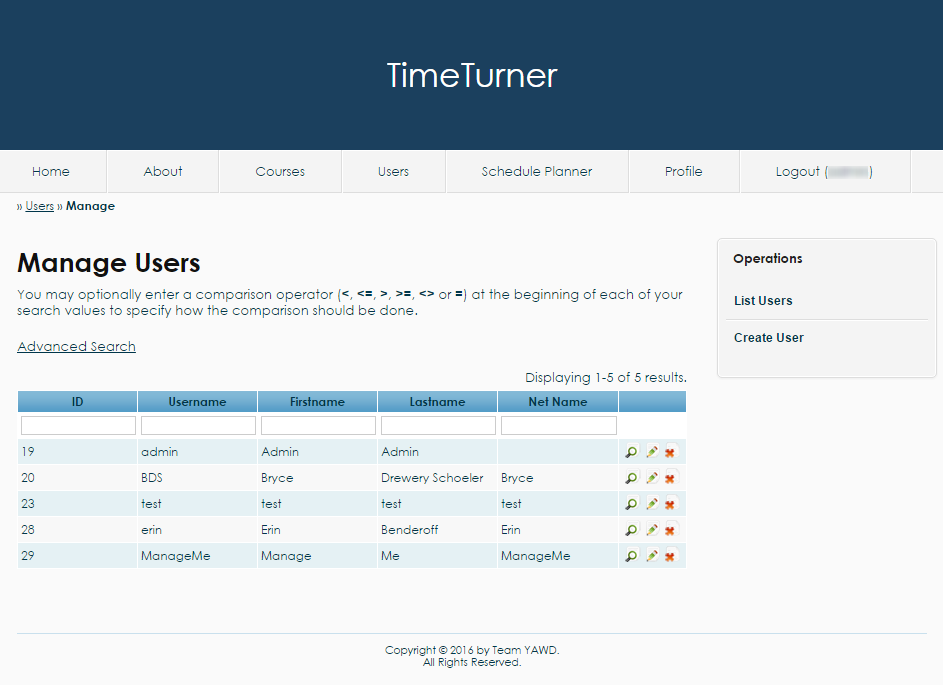
*Figure 4.2.40. ‘Users’ tab from the main menu when logged in as an Administrator*

The administrator will then see a list of users on the system, along with a list of operations that the administrator is allowed to perform. To access the user management page, the administrator must click on the ‘Manage Users’operation (see Figure 4.2.41).



*Figure 4.2.41. ‘Manage Users’ operation found on the ‘Users’ tab*

The administrator should then land on the user management page where multiple search operations are presented with a default search result containing a list of editable users (see Figure 4.2.42).

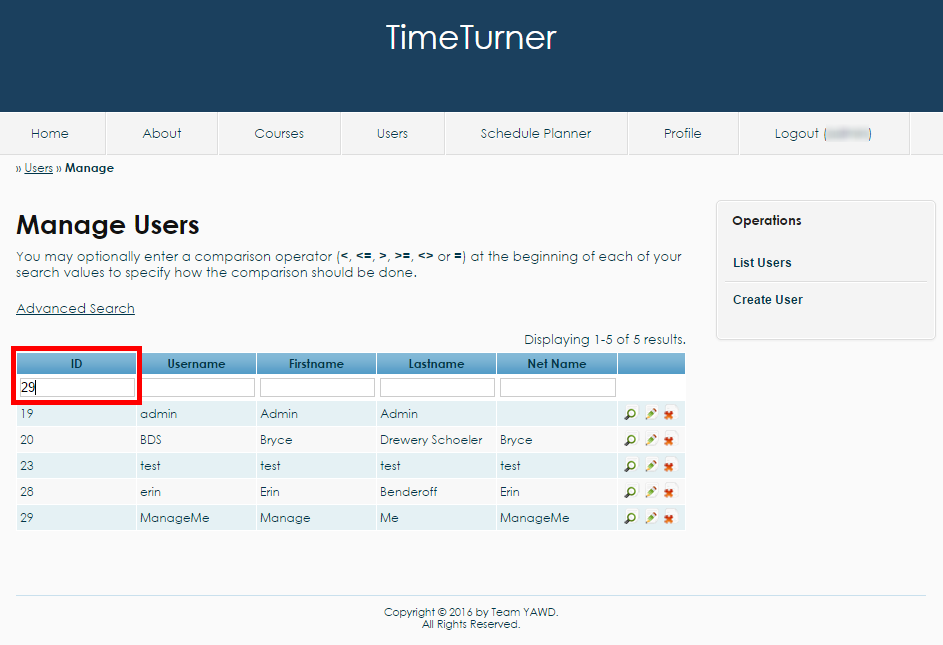


*Figure 4.2.42. User management page with a default table of users*

The following next operations will be performed from the user management page.

Search for Users

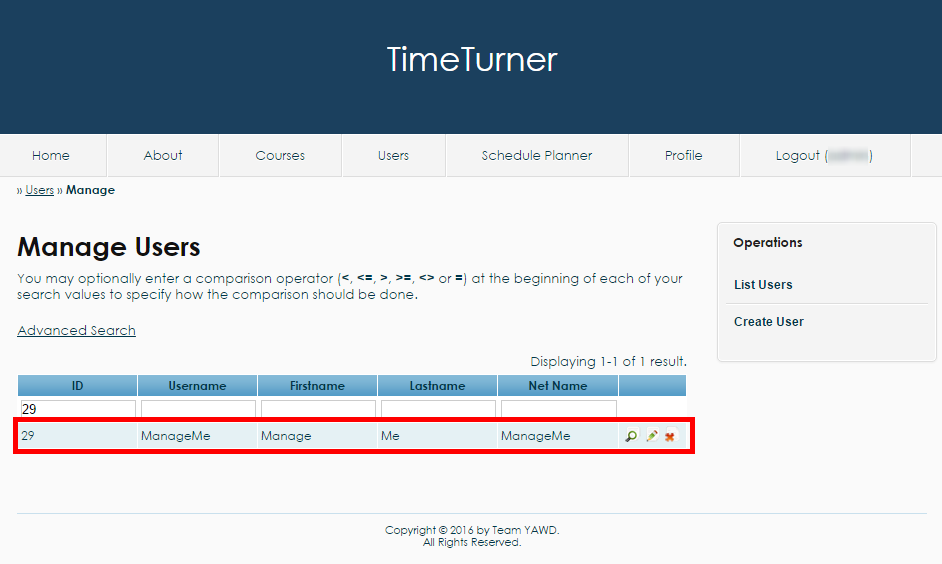
From the user management page, users can be found using the textboxes located on the second row of the table of users. For instance, if an administrator is looking for the user whose ID is 29, the ID number would be typed on the textbox in the ‘ID’ column (see Figure 4.2.43).



*Figure 4.2.43 Entering the desired search criteria in the corresponding textbox*

Once the field is completed, the administrator can press ‘enter’ or simply click outside of the textbox to refresh the search table. As a result, the user with the matching ID will be displayed on the table (see Figure 4.2.44). If no users have been found to match the search criteria, the table of users will be replaced with a message indicating that no results have been found.

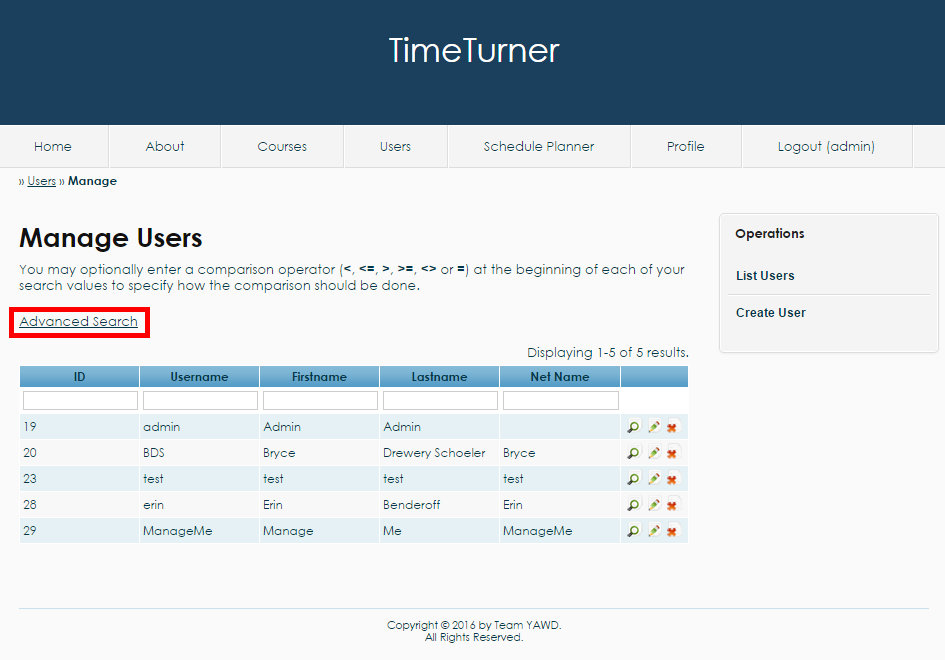
Note that many criteria can be entered in the various textboxes. In such case, the table will reload every time a textbox is out of focus or the key ‘enter’ is pressed. Also, comparison operators can be used in those textboxes, for example to filter all users having an ID less than 30. Another feature is that all search fields except for ID based on a ‘contains’ relationship, so for example, if we are looking for all usernames containing the letter ‘t’, ‘t’ could be written in the username search field. The search is also case insensitive.



*Figure 4.2.44 Result of the simple search by ID*

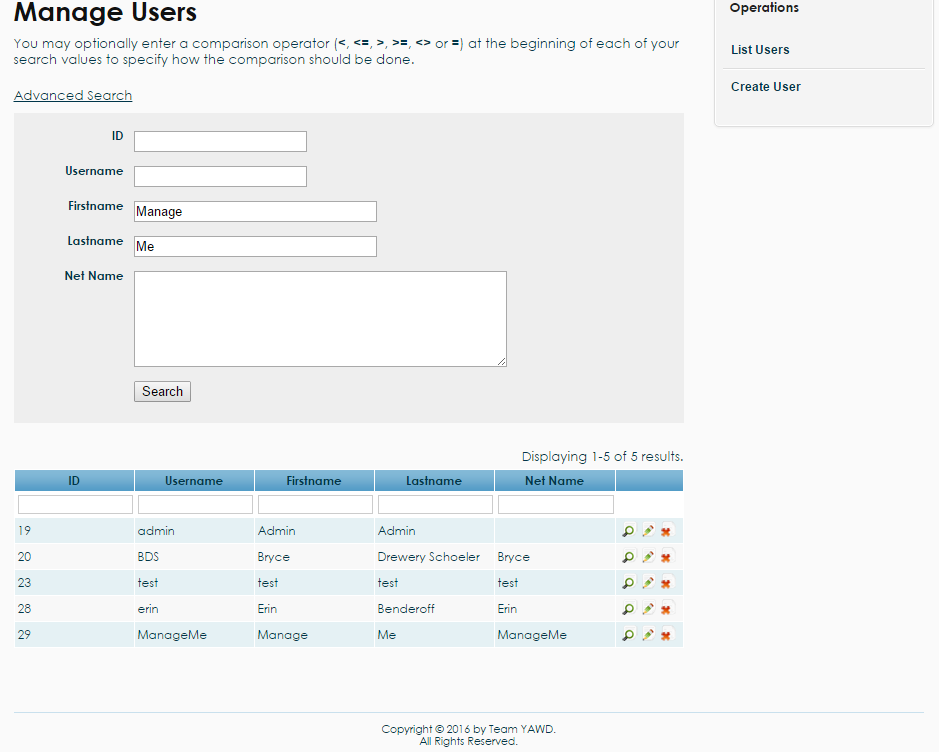
Advanced Search for Users

An administrator may decide to perform an advanced search if many criteria are to be used in the search, and the administrator does not wish the table to constantly refresh. To do so, the administrator may click on the ‘Advanced Search’ link (see Figure 4.2.45).

******

*Figure 4.2.45. The ‘Advanced Search’ link*

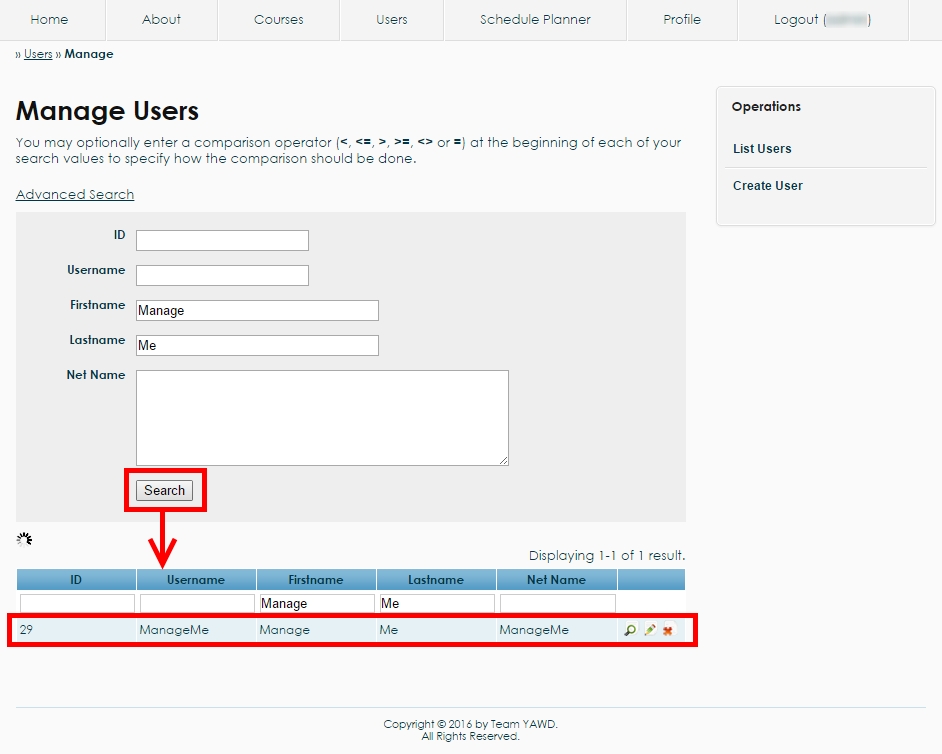
An advanced search form should appear once the link has been clicked. The form allows the administrator to look for a user using a combination of multiple criteria by filling up the fields that the administrator wishes to use (see Figure 4.2.46).



*Figure 4.2.46 Usage of the advanced search fields*

Once all the criteria have been entered, the administrator can hit the ‘search’ button at the bottom of the advanced search form. A table should be loaded with the user(s) that match the criteria (see Figure 4.2.47). Notice also that the search criteria have been copied into the textboxes for the simple search for clarity.

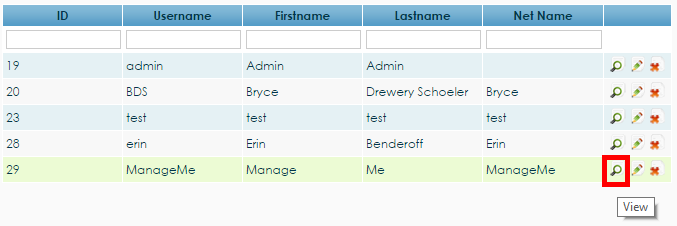
Like the simple search, the advanced search textboxes support comparison operators and will also display a message that no results have been found if the system fails to find a user based on the specified criteria. The searches are also based on a ‘contains’ relationship, meaning that all users which search field(s) contain the search criteria will be displayed. The text entered in the search fields are case insensitive.



*Figure 4.2.47. Post-advanced search results*

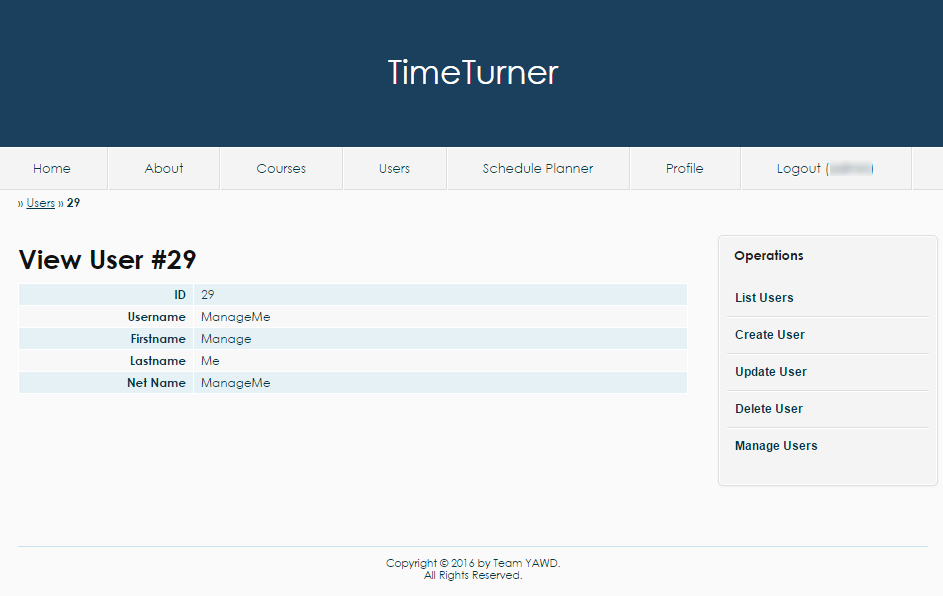
View User

Whenever a table of users is displayed, the administrator may decide to view the profile of a user. To consult the profile of a user, the administrator may click on the magnifying glass found on the right of row corresponding to the user (see Figure 4.2.48).



*Figure 4.2.48. The view magnifying glass corresponding to the user 29 allowing administrators to consult the profile of this user*

Once the magnifying glass has been clicked, the profile of the user should be displayed among with a list of available operations that will be discussed in the following sections (see Figure 4.2.49).

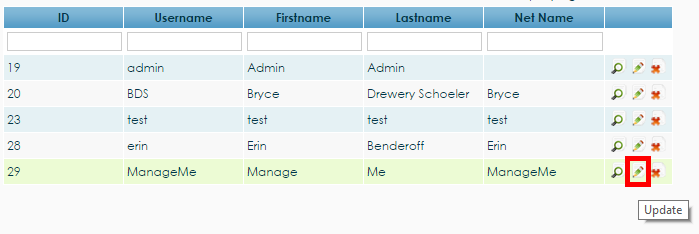


*Figure 4.2.49 The profile of a user with a set of operations on the right*

Update User

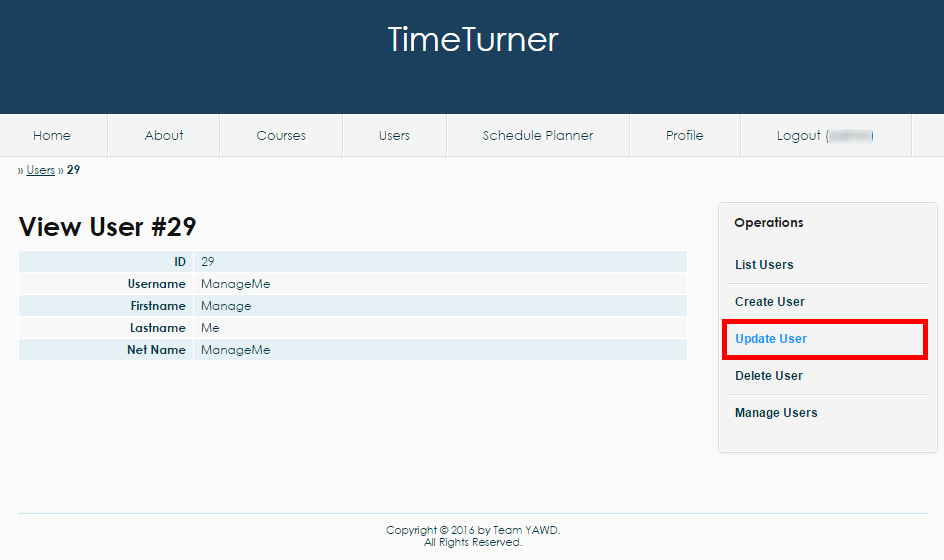
Users’ information can be updated through the user management page. More precisely, updating users’ information can be either be done through the table of users, or through consulting the profile of a user.

To update a user’s information through the table of users, the pencil button found on the right of the row corresponding to the user to be updated should be clicked (see Figure 4.2.50).



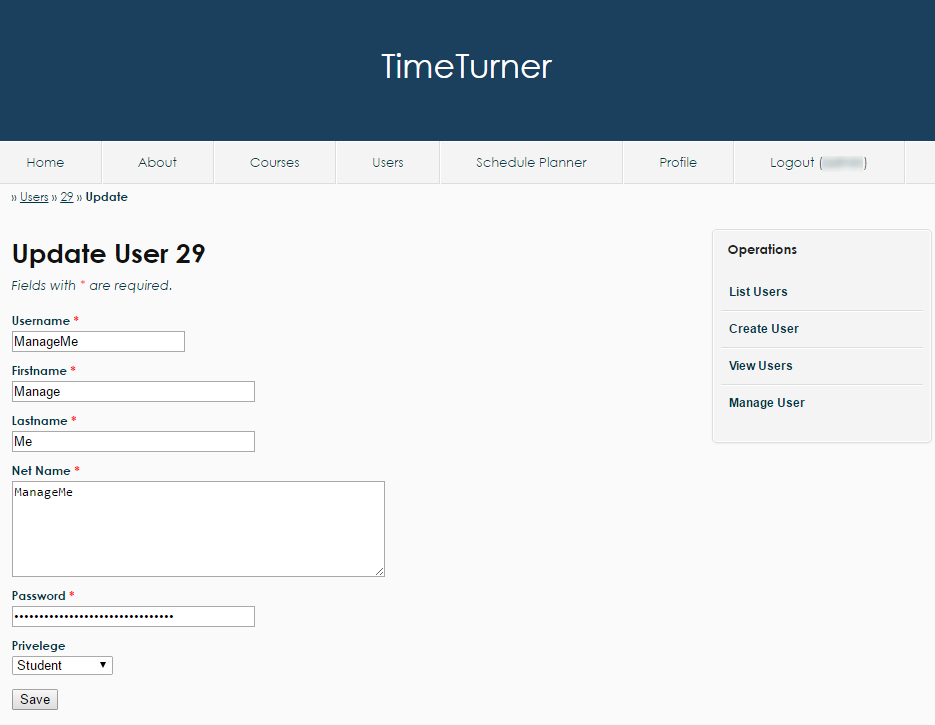
*Figure 4.2.50 Pencil button to update the user 29*

To update a user’s information through the user’s profile view, the administrator needs to choose the ‘Update User’ operation from the list of operations found on the left of the page (see Figure 4.2.51).



*Figure 4.2.51 The operation ‘Update User’ found on the user’s profile page view by an administrator*

From either method, the ‘Update User’ page should now be displayed. The page contains a form which fields are completed with the user’s current information. The administrator can decide to change any of those field and save the changes by clicking the ‘Save’ button (see Figure 4.2.52).



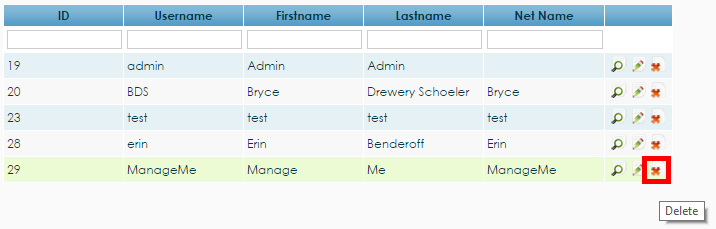
*Figure 4.2.52. The ‘Update User’ page having fields that contain the user’s initial information and that can be modified by the administrator once the ‘Save’ button has been clicked*

Once the save button has been clicked, the updated profile of the user should be displayed. Otherwise, some conditions about the fields might have been violated and the administrator needs to correct those violation before updating the user.

Delete User

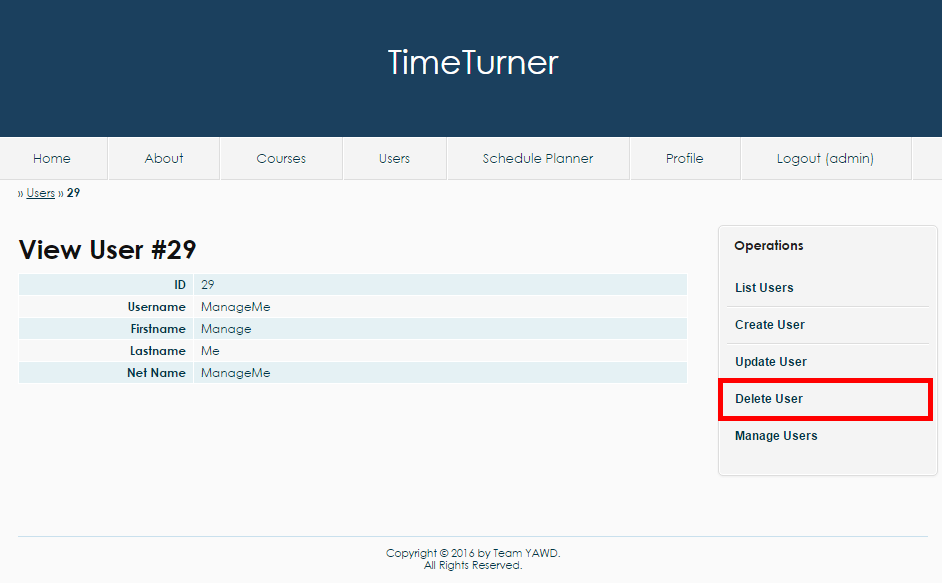
The administrator also has the power to delete a user through the user management page. Once again, deleting a user can be done either through the table of users or through the profile of the user.

To delete a user using the table of users, the administrator should click the ‘X’ button found on the same row as the user to be deleted (see Figure 4.2.53).



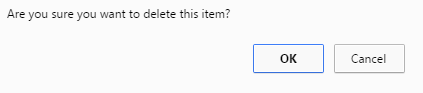
*Figure 4.2.53. The ‘X’ button to delete user 29*

To delete a user through the user’s profile page, the operation ‘Delete User’ should be clicked from the list of operations found on the right (see Figure 4.2.54).



*Figure4.2.54. The operation ‘Delete User’ on the user’s profile page viewed by an administrator*

When either the ‘X’ button or the ‘Delete User’ operation is clicked, a message should pop up to confirm the administrator’s decision (see Figure 4.2.55).

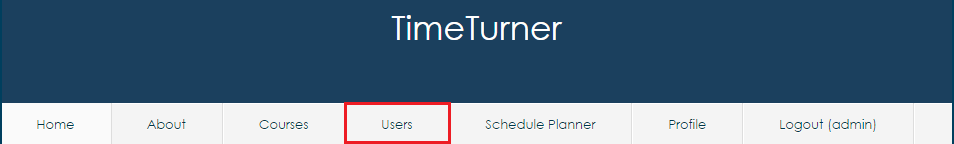


*Figure 4.2.55. Confirmation by the system to delete a user*

The administrator should press the ‘OK’ button on the confirmation box to delete the user. Once deleted, the administrator will be brought back to the user management page with an updated table of users.

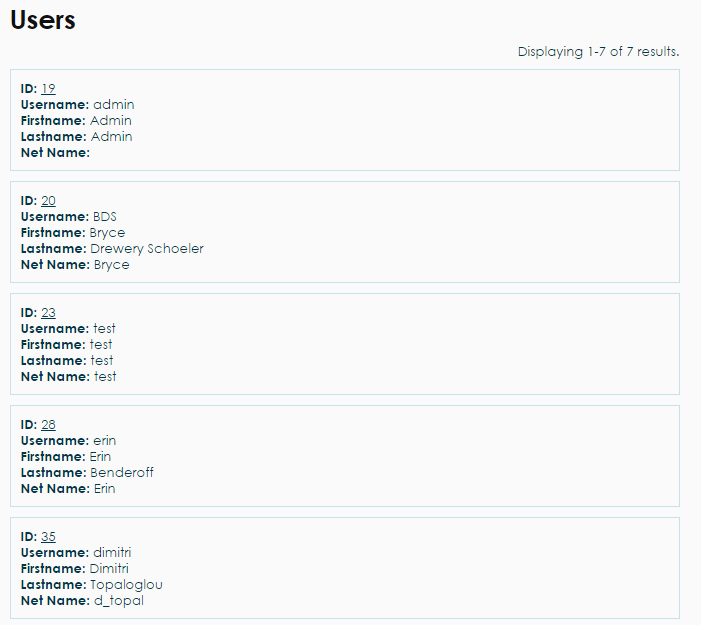
Viewing Users

An administrator can view all users currently in the database. To do so, he/she can select the user’s tab from the home page, once they have logged in (shown in Figure 4.2.56)



*Figure 4.2.56: Accessibility to the ‘User’ tab from the home screen*

Once the tab has been selected, all the users will be displayed on screen. The users are sorted by ID and displayed are each user’s specific credentials, such as, first and last name and their net names (shown in figure 4.2.57)



*Figure 4.3.57. All users displayed on the ‘User’ tab*

# Final Cost Estimation

Final Estimations

Deliverable 0 Estimation

|  |  |  |  |
| --- | --- | --- | --- |
| **Task Names** | **Cost (Hours)** | **Revised Cost (Hours)** | **Final Cost (Hours)** |
| Domain Model | 6 | 6 | 6 |
| Identification of Technologies | 12 | 12 | 12 |
| Team Members | 0.5 | 0.5 | 0.5 |
| Implementation Discussion | 5 | 5 | 5 |
| DMO Descriptions | 6 | 6 | 6 |
| **Total** | **29.5** | **29.5** | **29.5** |

Deliverable 1 Estimation

|  |  |  |  |
| --- | --- | --- | --- |
| Task Names | Cost (Hours) | Revised Cost (Hours) | Final Cost (Hours) |
| Requirements | 10 | 10 | 36 |
| Use Case Diagrams/Models | 8 | 8 | 20 |
| Resources | 5 | 5 | 8 |
| Scope | 5 | 5 | 8 |
| Architecture | 7 | 7 | 15 |
| Learning Technologies | ---- | ---- | 137.85 |
| **Total** | **35** | **35** | **224.85** |

Deliverable 2 Estimation

|  |  |  |  |
| --- | --- | --- | --- |
| Task Names | Cost (Hours) | Revised Cost (Hours) | Final Cost (Hours) |
| Programming Architecture | 100 | 116.4 | 125.5 |
| Subsystem Specification | 50 | 33.95 | 43.95 |
| Dynamic Design Scenarios | 25 | 29.1 | 30 |
| Estimation | 15 | 9.7 | 8.44 |
| Prototyping and Risk | **----** | 4.85 | **5** |
| **Total** | **190** | **194** | **212.89** |

Deliverable 3 Estimation

|  |  |  |  |
| --- | --- | --- | --- |
| Task Names | Cost (Hours) | Revised Cost (Hours) | Final Cost (Hours) |
| Execute Test Cases | 30 | 29.1 | 29.1 |
| Debugging/Optimizing code based on test cases | 15 | 29.1 | 30 |
| User Performance Test | 20 | 33.95 | 35 |
| Security and Stress Testing | 10 | 4.85 | 4.47 |
| **Total** | **75** | **97** | **98.57** |

Deliverable 4 Estimation

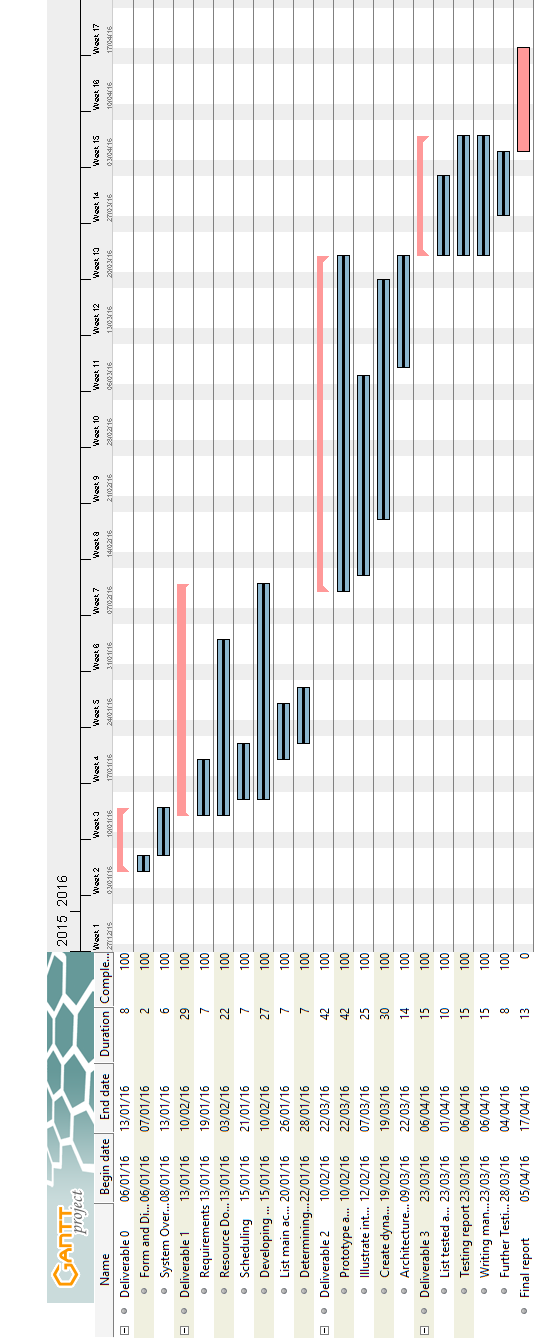
|  |  |  |  |
| --- | --- | --- | --- |
| Task Names | Cost (Hours) | Revised Cost (Hours) | Final Cost (Hours) |
| Review and Edit past work | 20 | 29.1 | 29.1 |
| Compiling data into one file | 30 | 9.7 | 9.7 |
| Final Test on System | 10 | 9.7 | 9.7 |
| **Total** | **60** | **48.5** | **48.5** |

Final Estimations

|  |  |  |  |
| --- | --- | --- | --- |
| **Deliverable Names** | **Cost (Hours)** | **Revised Cost (Hours)** | **Final Cost (hours)** |
| Deliverable 0 | 29.5 | 29.5 | 29.5 |
| Deliverable 1 | 35 | 35 | 224.85 |
| Deliverable 2 | 190 | 194 | 212.89 |
| Deliverable 3 | 75 | 97 | 98.57 |
| Deliverable 4 | 60 | 48.5 | 48.5 |
| **Total** | **389.5** | **404** | **614.31** |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Claudia | Dimitri | Daniel | Lori | Erin | Bryce | Aline | Philip | Ideawin | Ryan | Marc | Kevin |
| Deliverable 1 | 19.3 | 27 | 20 | 22 | 19 | 20 | 16.4 | 16.15 | 14 | 16 | 20 | 15 |
| Deliverable 2 | 23.06 | 34 | 22 | 11.6 | 11.5 | 15 | 19.6 | 12.13 | 15 | 20 | 15 | 14 |
| Deliverable 3 | 9.7 | 8 | 12 | 11.3 | 14.67 | 10 | 5.4 | 8.3 | 7 | 5 | 10 | 5.5 |

Updated Gantt Chart



*Figure 5.2.1 Updated Gantt chart*