



## Department of Computer Science & Software Engineering

Software Process: SOEN341/4S---2016

### Testing & Delivery

The Schedulers	
Emili Vasseva	27526741
Sean Marcoux	27511876
Bruce Edouard Brazier	27419562
Dias Marat	27277911
Le Vinh Dang	26844987
Adriel Fabella	27466005
Gabriele Bavaro	27399103
Ying-Chen Chu	27415710
Alex Eladas	27041462
Salma Aly	27176414
Adil Hssaini	24832396
Nicolas Frazer-McKee	27068956

## Table of Contents

Grading Sheet.....	
1. Introduction.....	1
2. Testing Report.....	1
2.1 Test Coverage .....	1
2.1.1 Tested Items.....	2
2.1.2 Untested Items of Interest.....	2
2.2 Test Cases .....	3
2.2.1 Unit Testing .....	3
2.2.2 Requirements Testing .....	10
2.2.3 Stress Testing.....	20
2.2.4 Security Testing.....	27
3. System Delivery .....	28
3.1 Installation Manual .....	28
3.2 User Manual.....	40
3.3 Administrator Manual.....	47
4. Final Cost Estimate .....	48

## *Grading Sheet*

<i>Section</i>	<i>Evaluation criteria (see instructions in the template for details)</i>	<i>Grading</i>
<i>all</i>	<i>10 marks are allocated for excellence, professionalism and quality of work above and beyond the correct meeting of specifications.</i>	<i>/10</i>
<i>1</i>	<i>Presentation of the document</i>	<i>/5</i>
<i>2</i>	<i>Introduction</i>	<i>/1</i>
<i>3.1</i> <i>3.2</i>	<i>Completeness of covered/uncovered items. Rationale of the importance of testing these items.</i>	<i>/3</i>
	<i>Testing</i>	
	<i>Reproducibility of test cases. Exact description of test input data and expected results, and the procedure to convey all test cases. Description of the rationale for the derivation of each test case, e.g. equivalence partitioning analysis, branch coverage analysis, etc.</i>	
<i>3.2.1</i>	<i>Unit testing</i>	
<i>3.2.2</i>	<i>Requirements testing:</i>	<i>/4</i>
<i>3.2.3</i>	<i>Stress testing</i>	<i>/8</i>
<i>3.2.4</i>	<i>Security testing</i>	<i>/1</i>
		<i>/1</i>
<i>4.1</i> <i>.</i>	<i>Clarity of instructions. Self-inclusion of the installation procedure, i.e. the installation does not necessitate the installation of external resources.</i>	<i>/7</i> <i>.</i>
<i>4.2</i>	<i>Clarity of instructions. Completeness of instructions, i.e. all system features' usage instructions are provided.</i>	<i>/7</i>
<i>5</i>	<i>Completeness and clarity of cost to date in terms of person hours.</i>	<i>/3</i>
<i>Total</i>		<i>/50</i>

*DO NOT REMOVE THIS PAGE WHEN SUBMITTING YOUR DOCUMENT*

# 1. Introduction

The present deliverable details the testing and delivery processes that were undertaken to the completion of the scheduling system. In its first part it deals with the testing coverage in which all test items along with the test cases that were applied on them will be listed, as well as an explanation as to why testing is required. Two testable units are identified. The first one is the authentication which includes the log in and registration units, and the second one is the “manage Preferences” unit. Both were selected because of their priority and relevance to the overall system. A list of test cases is included for each unit in addition to the code to the stubs and drivers used. Extra tests are performed to show how the system is to be used and what system reactions are to be expected.

Stress and Security testing follow to evaluate the robustness and security levels of the system. Those tests shed light on the potential extreme situations of system usage and on its resistance to security concerns such as SQL injection attacks.

The report extensively explains the installation process. A manual listing what is needed and how it is to be installed is made available, and it guarantees a successful and functional system upon completion of the provided steps. It also describes how the system is to be used along with all its available features.

The report concludes with a revised cost estimate that compares what was previously forecasted and what was actually incurred.

## 2. Testing Report

### 2.1 Test Coverage

The following section covers the testing phase of the project. Testing is a key concept when it comes to the delivery of a particular product. The reason being is that it allows for the workers to assess the early problems in the software and to ensure that the requirements of top priority are managed, since these functionalities may endanger the overall software. The section is divided into different test categories:

- Tested Items
- Untested Items
- Unit testing
- Requirement testing

**Tested Items:** Defined as the features that were already tested. It results into two possible outcomes: pass or fail.

**Untested Items:** Defined as the features that were not already tested due to a specific reason which results into not prioritizing that particular matter.

**Unit Testing:** Defined as testing a small part of the software, in other words, a small part of the code is evaluated to see if it works.

**Requirements:** Defined as the essential functionality that a user, student and admin must be able to accomplish with the software at hand.

### **2.1.1 Tested items**

The two units which were tested on consist of the authentication unit and the preference unit.

**Authentication:** This was immediately tested since the requirement of signing up and logging forms the basis of the whole use of the software. This consists of ensuring that the user is successfully directed to his/her appropriate page. This item is the number one priority because this functionality allows access to the Scheduler.

**Manage Preferences:** This unit was tested because the integration of preferences demands to be of creative thinker when generating the schedule. This is of great importance because the system's requirement is to take into consideration the things that the student wants and testing solely on effectively containing, changing and deleting the preferences proves to be of big priority.

**Stress and Security:** This is an important key because the project, assuming it will be used by a certain client for a big academic institution, provides access to personal information. These tests are of importance since they provide a safety margin. Without protected and secure access, the software would suffer and, nonetheless the clients, since its functionality is overshadowed by not being secured and properly stressed.

### **2.1.2 Untested Items of Interest:**

**Remaining server calls:** There are server calls for obtaining all courses, sections, and user information as well as calls for updating all of that information and calls for generating the schedule. These calls can be unit tested the same way the other calls were. They are important to test because the user interface relies on the server calls behaving as expected. Whether they succeed or fail, they must do it in the way the UI would expect.

**The user interface:** This was user tested, but it was not tested programmatically because there is no official React unit testing framework. There are some open source frameworks that we did not look too much into because they are very new and are neither robust nor heavily tested. Unit testing a UI is also much more complicated than testing a single function that handles data. If we were to unit test, we would use one of the open source React unit testing frameworks and edit it as needed. It is important to test the UI because this is the user facing portion of the application. If this does not perform as expected, it is immediately apparent to the user and does not look professional. User testing does not always pick up on all of the bugs the same way a unit test does, so unit testing the UI would thoroughly verify that it performs as expected.

## **2.2. Test Cases**

### **2.2.1 Unit Testing**

For the unit testing we chose to test several important server calls. These are crucial because it is how the client side and server side of our application communicate. Without these working bug-free the application would not be able to do much.

All of the server calls interact with the database in some way, either to retrieve data or to update it. In order for testing to work consistently we needed a database that would be constant. For this we made a much smaller database purely for testing purposes. When running tests locally, this database will be used instead of the official one. That way we will know exactly what information is supposed to be stored in the database and can expect consistent results from our tests.

Laravel has an official unit testing framework, but it expects all of the mvc to be used. Since we replaced the view with React, this framework was not ideal. Instead we have our own testing script, coded in javascript that runs all of our tests. Neither the testing script nor the database will be included in the release version of our application, since they are only for testing.

For development we created a javascript object, `realServerBridge`, which we used to abstract all of the server calls. We used this to make the server calls when testing.

### **Login:**

Our login server call is as follows:

URL	Purpose	Type	Input	Returns
/login	Login verification	POST	{username:'', password:''}	{"success":"true","username": "JSONB","isAdmin":"true"}

It expects a JSON object including a username string and a password string. It will return a JSON object with a string holding the success state, the username, and a string saying whether or not the user is an admin. The test cases are as follows:

Description	Input	Expected output that was tested
Call is made with a username that exists in the database with the correct password for that user	{username: 'User', password: 'password'}	success='true' username='User' isAdmin='false'
Call is made with a username that does not exist in the database (sending an empty username and password is treated the same way as this)	{username: 'notouser', password: 'password'}	success='false'
Call is made with existing user, but the wrong password	{username: 'User', password: 'notthepassword'}	success='false'
Call is made with the admin's username and password	{username: 'Admin', password: 'password'}	success='true' username='Admin' isAdmin='true'

We tested by making the server calls and then verifying the output we received. No set up or tear down was needed to keep the test consistent.

### **Set Preferences**

Users can specify preferences for their schedule, such as a desired course load or a day of the week they want free. The server call to update the preferences for a user is as follows:

URL	Purpose	Type	Input	Returns
/editpreferences	Edit user's preferences	POST	{username:'',cloud:",day off:",pretime:"}}	{"success":"true","username":"JasonB","courseload":"5","dayoff":"Monday","pretime":"Mornings"}

It expects a JSON object including strings for the username, course load, day off, and preferred time and will return a JSON object including the success state, as well as the info that was sent. Our client side of our project can handle invalid data entered for preferences by replacing it with default information, so for testing we were not concerned with that. The test cases are as follows:

Description	Input	Expected result
Call is made with a username that exists	{username:'Jason', cloud:'5', dayoff:'Monday', pretime:'Mornings'}	Data updated in the database
Call is made with a username that is not in the database	{username:'notausser', cloud:'5', dayoff:'Monday', pretime:'Mornings'}	Output: success='false'

For the first test case we had to also use the server call to retrieve the preferences in order to check that the information had actually been updated.

The methods for both cases expect a username in the cookies, so this had to be done as set up for the cases. This had to be removed for the tear down and the preferences had to be set back to blank values, so the tests could be consistent.

### **Registration:**

Registration was important to test. If this didn't work it could cause security issues or could make it so that people could not even sign up for the site and could never access it. The server call is as follows:

URL	Purpose	Type	Input	Returns
/register	Register user	POST	{username:'', email:'', password:''}	{"success":"false","username":"SprinkKing","error":"username already taken"}



It expects a JSON object containing strings for the username, e-mail, and password and will return a JSON object containing strings for the success state, the username, and an error message. The success will be false only if a user with that username already exists. The error will be blank if the call was successful.

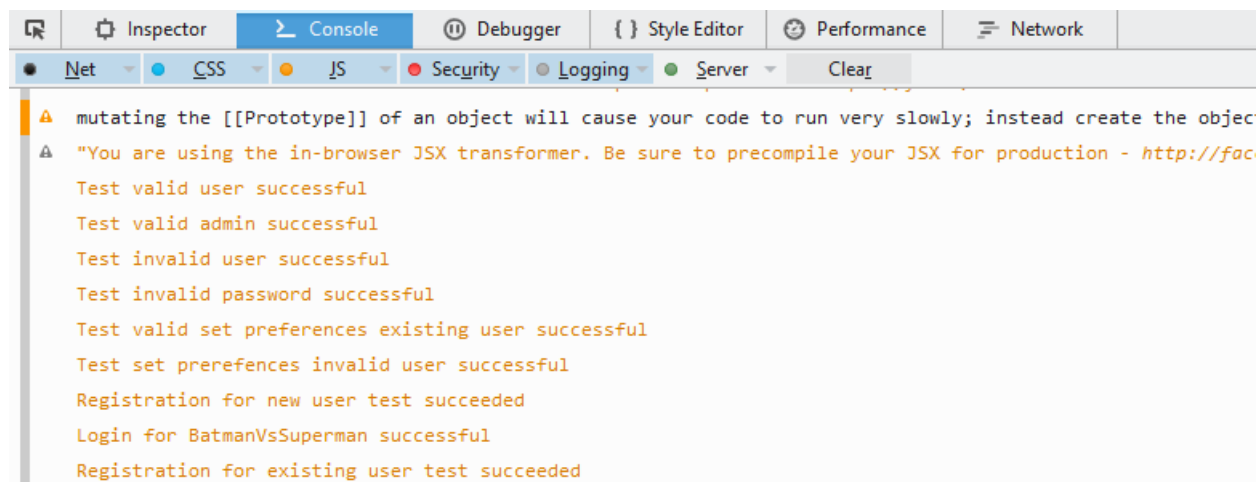
The test cases are as follows:

Description	Input	Expected output that was tested
Call is made with a new username	{username:'BatmanVsSuperman', email:'justiceleague@gmail.com', password:'password'}	success='true'  Also the following login call should succeed, showing that the user was added in the database
Call is made with an existing username	{username:'Jason', email:'jason@hotmail.com', password:'password'}	success=false

No set ups were needed. For tear down we had to remove the BatmanVsSuperman user from the database, so the test would work again next time.

## Running the tests

All of the tests succeeded. Here was the console after they ran:



The screenshot shows a web browser's developer console with the 'Console' tab selected. The console displays several messages, including warnings about mutating the prototype and using the in-browser JSX transformer. Below these, a series of test results are shown in orange text, indicating that all tests passed successfully. The tests include: 'Test valid user successful', 'Test valid admin successful', 'Test invalid user successful', 'Test invalid password successful', 'Test valid set preferences existing user successful', 'Test set preferences invalid user successful', 'Registration for new user test succeeded', 'Login for BatmanVsSuperman successful', and 'Registration for existing user test succeeded'.

```

mutating the [[Prototype]] of an object will cause your code to run very slowly; instead create the object
You are using the in-browser JSX transformer. Be sure to precompile your JSX for production - http://facebook.github.io/react/docs/using-jsx-with-babel.html#example-with-babel
Test valid user successful
Test valid admin successful
Test invalid user successful
Test invalid password successful
Test valid set preferences existing user successful
Test set preferences invalid user successful
Registration for new user test succeeded
Login for BatmanVsSuperman successful
Registration for existing user test succeeded

```

## Unit Testing Code

For any server calls that expect information back from the server, a callback method must be sent to the server bridge object. This is due to the asynchronous nature of AJAX calls. When the data is finally

received, it will be sent as the only argument to that method and then it can be tested. This is why we send functions as an argument for the server bridge methods.

```
var serverBridge=realServerBridge;
//LOGIN TESTS
var response = serverBridge.login('User', 'password', function(data) {
  if(data.success=='true'&&data.username=='User'&&data.isAdmin=='false') {
    console.log('Test valid user successful');
  }
  else {
    console.log('Test valid user failed');
  }
});
response = serverBridge.login('Admin', 'password', function(data) {
  if(data.success=='true'&&data.username=='Admin'&&data.isAdmin=='true') {
    console.log('Test valid admin successful');
  }
  else {
    console.log('Test valid admin failed');
  }
});
response = serverBridge.login('notuser', 'password', function(data) {
  if(data.success=='false') {
    console.log('Test invalid user successful');
  }
  else {
    console.log('Test invalid user failed');
  }
});
response = serverBridge.login('User', 'notthepassword', function(data) {
  if(data.success=='false') {
    console.log('Test invalid password successful');
  }
  else {
    console.log('Test invalid password failed');
  }
});
```

```

//SET PREFERENCES TESTS
//Set up
var testPref = {"courseLoad":"5", "day":"Monday", "time":"Mornings"};
var response = serverBridge.editPreferences(testPref);
var cookie = cookieManager.addCookie("username", "Jason", 1);
//Test
serverBridge.getUserPrefs(function(data) {
    if (data.courseLoad == testPref.courseLoad && data.dayoff == testPref.day && data.preferredTime == testPref.time){
        console.log('Test valid set preferences existing user successful');
    }
    else{
        console.log('Test valid set preferences existing user failed');
    }
});
//Tear down
serverBridge.editPreferences({"courseLoad": "", "day" : "", "time" : ""});
cookieManager.removeCookie("username");

//Set up
cookieManager.addCookie('username', 'notausser', 1);
//Test
var response = serverBridge.editPreferences(testPref, function(data) {
    if(data.success=='false') {
        console.log('Test set preferences invalid user successful');
    }
    else {
        console.log('Test set preferences invalid user failed');
    }
});
//Tear down
cookieManager.removeCookie("username");

```

```

//REGISTRATION TESTS
response = serverBridge.register('BatmanVsSuperman', 'justiceleague@gmail.com', 'password', function(data) {
    if(data.success == "true"){
        console.log("Registration for new user test succeeded");
    }
    else{
        console.log("Registration for new user test failed");
    }
});
var response = serverBridge.login('BatmanVsSuperman', 'password', function(data) {
    if(data.success=='true'&&data.username=='User'&&data.isAdmin=='false') {
        console.log('Login for BatmanVsSuperman successful');
    }
    else {
        console.log('Login for BatmanVsSuperman failed');
    }
});

response = serverBridge.register('Jason', 'jason@hotmail.com', 'password', function(data) {
    if(data.success == "false"){
        console.log("Registration for existing user test succeeded");
    }
    else{
        console.log("Registration for existing user test failed!");
    }
});

```

### **2.2.2 Requirements testing**

The tables below show the result of black box testing for the functional requirement of the schedule generator system. Cells highlighted in green indicate a passed test while the ones in red indicates a failure. Failures are currently being investigated to be fixed for the final deliverable v1.2

Cases that were scoped out and therefore not tested are indicated in the last table

UC1	Login				
ID	Description	Expected Output	Result	Bug ID	Comments
1.1	User inputs valid username and password (matching inputs)	User is redirected to student main page	PASS		
1.2	User inputs a valid password and "admin" username	User is redirected to admin main page	PASS		
1.3	User inputs invalid username and/or password	"Incorrect password and/or username"	PASS		
1.4	User tries to login with one of the fields empty	"Incorrect password and/or username"	PASS		

UC2	Logout				
ID	Description	Expected Output	Result	Bug ID	Comments
2.1	User clicks on "logout" button	User is redirected to student main page	PASS		

UC3	Reset Password				
ID	Description	Expected Output	Result	Bug ID	Comments
3.1	User clicks on "reset password" option on the login page	An email is sent asking user to confirm by clicking on a link to reset the password	FAIL	1	To be fixed for v1.2

UC4	Sign up				
ID	Description	Expected Output	Result	Bug ID	Comments
4.1	User inputs valid username, email and password.	Account is created and the user is redirected to the preferences page in their new account	PASS		
4.2	User inputs a valid password and username, but an invalid email.	Error Message displayed asking user to enter a valid email	FAIL	2	To be fixed for v1.2
4.3	User inputs a valid password and email, but an invalid username.	Error Message displayed asking user to input a username with 4-6 characters	PASS		
4.4	User inputs "admin" as a username, with a valid password and email.	Error message appears notifying user that the username entered is invalid	PASS		

UC7	View auto-generated list of taken courses				
ID	Description	Expected Output	Result	Bug ID	Comments

7.1	User inputs a valid number of semesters taken (between 1 and 7) and selects “generate course list”	A pop-up asking for confirmation to generate the course list is displayed.	PASS		
7.2	User selects “Yes” in the pop-up prompting for confirmation to generate the course list	The pop-up closes and courses are auto-generated and added to the list of taken courses	PASS		
7.3	User selects “No” in the pop-up prompting for confirmation to generate the course list	The pop-up closes and the input to “semesters taken” is blank.	PASS		
7.4	User inputs an invalid number of semesters taken (not between 1 and 7) selects “generate course list”	An error message is displayed and the “generate course list” option doesn’t do anything.	PASS		

UC8	Set taken courses				
ID	Description	Expected Output	Result	Bug ID	Comments
8.1	User selects the “add class” option for the taken courses list	User is presented with a pop-up box prompting for an input of a course number or an input of a course name	PASS		
8.2	User inputs a valid (present in the database) course number and selects “add class”	The course name corresponding to the course number is automatically generated. The pop-up box closes, and the inputted course is added to the list of taken courses	PASS		

8.3	User inputs a valid (present in the database) course name and selects “add class”	The course number corresponding to the course name is automatically generated. The pop-up box closes, and the inputted course is added to the list of taken courses	PASS		Fixed v1.1
8.4	User inputs a valid (present in the database,) course number and/or course name already in the list of taken courses and selects “add class”	An error message is displayed and selecting the “add class” option doesn’t do anything.	PASS		

UC9	Set needed courses				
ID	Description	Expected Output	Result	Bug ID	Comments
9.1	User selects the “add class” option for the needed courses list	User is presented with a pop-up box prompting for an input of a course number or an input of a course name	PASS		
9.2	User inputs a valid (present in the database, does not conflict with another course time and the prerequisites are met) course number and selects “add class”	The course name corresponding to the course number is automatically generated. The pop-up box closes, and the inputted course is added to the list of needed courses	PASS		Fixed v1.1
9.3	User inputs a valid (present in the database, does not conflict with	The course number corresponding to the course name is			



	another course time and the prerequisites are met) course name and selects “add class”	automatically generated. The pop-up box closes, and the inputted course is added to the list of needed courses	PASS		
9.4	User inputs a course number or course name that conflicts with the time of another course in the list of needed courses and selects “add class”	An error message is displayed and selecting the “add class” option doesn’t do anything.	FAIL	3	The check condition is not yet properly implemented  To be fixed for v1.2
9.5	User inputs a course number or course name for which the prerequisites are not met and selects “add class”	An error message is displayed and selecting the “add class” option doesn’t do anything.	FAIL	3	The check condition is not yet properly implemented
9.6	User inputs an invalid (not present in the database) course number and/or course name and selects “add class”	An error message is displayed and selecting the “add class” option doesn’t do anything.	PASS		
9.7	User inputs a valid (present in the database, does not conflict with another course time and the prerequisites are met) course number and/or course name already in the list of needed courses and selects “add class”	An error message is displayed and selecting the “add class” option doesn’t do anything.	PASS		Fixed v1.1

UC10	Delete needed courses				
ID	Description	Expected Output	Result	Bug ID	Comments
10.1	User selects the delete icon of a course in the list of needed courses	The course is removed from the list of needed courses	PASS		

UC11	Delete Taken Courses				
ID	Description	Expected Output	Result	Bug ID	Comments
11.1	User selects the delete icon of a course in the list of taken courses	The course is removed from the list of taken courses	PASS		

UC12	Set Preferences: Related to (Modify Preferences)				
ID	Description	Expected Output	Result	Bug ID	Comments
12.1	User selects his desired day off to be a specific day (Monday, Tuesday, Wednesday, Thursday, or Friday) from the drop down menu	The selected day is displayed as the desired day off	PASS		
12.2	User selects his desired day off to be “none” from the drop down menu	“none” is displayed as the desired day off	PASS		
12.3	User selects his preferred time of the day to be either “mornings”, “afternoons” or “evenings” from the drop down menu	The selected preferred time of the day is displayed as the preferred time of the day	PASS		

12.4	User selects his preferred time of the day to be “Any” from the drop down menu	“Any” is displayed as the preferred time of the day	PASS		
------	--------------------------------------------------------------------------------	-----------------------------------------------------	------	--	--

UC13	Generate schedule				
ID	Description	Expected Output	Result	Bug ID	Comments
13.1	User selects the “build schedule” option	User is directed to the schedule page, and multiple schedules are generated according to the preferences and needed courses.	FAIL	4	The scheduling algorithm does not yet properly generate schedules according to the inputs (preferences and needed courses) given
13.2	User selects an arrow on the schedule page to see a different generated schedule	Another generated schedule is displayed on the schedule page	PASS		Fixed v1.1
13.3	User selects “select this schedule” from the schedule page	User is directed to the sequence page. The selected schedule is displayed along with the user’s course sequence	FAIL	4	The user’s course sequence does not display the right data

UC19	Edit Account Information				
ID	Description	Expected Output	Result	Bug ID	Comments

19.1	User changes the username by inputting a new valid username	Name is changed in the account management page	PASS		
19.2	User tries to change the username by inputting the same current user name	Error Message notifying the user that the entered user name is the same as the old one	PASS		Fixed v1.1
19.3	User changes the username with a new user name that is too short (less than 4 characters)	Error message notifying the user that the username is too short	PASS		
19.4	Change current email with a new valid email	New email appears on the user account information	PASS		
19.5	Change current email with one that has invalid format	Error message appears asking the user to input a valid email address.	FAIL	2	
19.6	Change Password by entering the correct current password and new password with 8-16 characters and retype the new password correctly	Password is changed	PASS		
19.7	Change Password by inputting the same current password as the new one	Error message appears notifying user to enter a new password	PASS		
19.8	Change password by entering the correct current one , entering a valid new one but mismatch in retyping the new one	Error message notifying the user that the passwords do not match	PASS		

UC23	Add Course to Program (Administrator)				
ID	Description	Expected Output	Result	Bug ID	Comments
23.1	Add a new course to list of courses offered	Course is added to the list of courses displayed	PASS		
23.2	Add a course that already exists in the list of courses	Error message appears and Course List is not changed	PASS		
23.3	Add Course with wrong name format ex: (341soen)	Error message appears and Course List is not changed	FAIL	5	

UC24	Edit Course (administrator)				
ID	Description	Expected Output	Result	Bug ID	Comments
24.1	Edit Course Description	Course description is changed	PASS		
24.2	Edit Course Prerequisites	Course prerequisites changes	PASS		

UC25	Delete Course (administrator)				
ID	Description	Expected Output	Result	Bug ID	Comments
25.1	Delete an existing course from course list	Course is removed from the course list	PASS		
25.2	Delete a course with ID not in the list	Error message appears and Course List is not changed	PASS		

UC26	Add Section (administrator)				
ID	Description	Expected Output	Result	Bug ID	Comments
26.1	Create new section with valid inputs (ex:Name:UI, Location: H-563 Day:MW Time:16:15-17:30,Course Name:SOEN341,Semester: W2016)	New Section is added to the list of sections of the required course	PASS		
26.2	Create Section without inputting all the data	Error message appears notifying the user to complete the data required	PASS		
26.3	Add Section with existing section name	Error Message appears notifying the admin that section already exists	PASS		
26.4	Add section with the same date, time and location of an existing section	Error Message appears to notify admin that the section cannot be created	PASS		

UC27	Edit Section (administrator)				
ID	Description	Expected Output	Result	Bug ID	Comments
27.1	Edit the time of an existing section (ex: SOEN 341, Time:18:00-19:00)	Section time is changed; new time appears in the section information	FAIL	6	To be fixed for v1.2

27.2	Edit the location of an existing section	Section location is changed; new location appears in the section information	FAIL	6	To be fixed for v1.2
------	------------------------------------------	------------------------------------------------------------------------------	------	---	----------------------

UC 28	Delete Section (administrator)				
ID	Description	Expected Output	Result	Bug ID	Comments
28.1	Delete an existing section from the list of sections of a course	Section is removed from the list	PASS		Fixed v1.1

**Requirements Scoped out and therefore not tested:**

Use Case Number	Description	Test Result
UC 5	View Default Schedule without creating an account	N/A
UC 6	Modify a generated schedule	N/A
UC 14	Save a generated schedule	N/A
UC 15	Delete a saved schedule	N/A
UC 16	Print a generated schedule	N/A
UC 17	View previously saved schedule	N/A
UC 18	Change a section in the schedule	N/A

### **2.2.3 Stress Testing**

The system described throughout the scheduler's deliverables is a system which, by the very nature of its usage, will experience varying degrees of client traffic: There will be varying volume of users logging in to the system during the day versus the middle of the night or the dates of class registration versus other days between or in the middle of any given semester. This volume of users will tend to be condensed during the same higher traffic times (i.e. during the day of course registration) and more spread out during other times. These periods of heavy load on the system can be simulated and

exaggerated beyond normal operation the system with a script efficiency tool. The resulting stress test can be applied to a system in order to determine the robustness, response and availability of a correct behavior for any system that may experience variable operation. The generic test is a black box test that can be applied to any system algorithm. However, the particular programming language and methods are a good indication

For the purpose of this deliverable, Apache Benchmark (ab) along with a PHP code in order console in order to repeat multiple http was used as a dynamic software verification and validation method in order to produce repeatable testing with quantitative request times (connect, processing, waiting, and total), transferred bytes (total, HTML, and document) for a given number of requests to be performed and the number of requests that occur simultaneously. A greater number of requests may represent simply a longer period of time (a week vs an hour). However, if a greater number of these requests occur at the same time (concurrency) than what is normal, then we are implementing a stress test in which we can observe several of the stress related defects.

The dynamic apachebenchmark test the PHP code and the server by attributing a number `-n` of requests and a value `-c` for the amount of simultaneous requests to be performed on every component of the server. This test however simulates a single device sending requests; in order to simulate many more devices, each with their own number set of requests, a PHP code was added to the apachebenchmark tool in order to multitask a number `-r` of url request repetition performed by each client and their value `-c` for the list of simultaneous clients.

Thus we have `-c` concurrent clients requesting the url's `-r` times each making each their `-n` requests with `-c` multiples.

This test is much more likely to lead to a failure of the code and an incident once that code runs causing an incident that may or may not be handled because of the larger number of simultaneous and total requests.

The PHP classes are: ezab and abrunner

```
class eZAB
{
    static $version = '0.3-dev';
    static $defaults = array(
        // 'real' options
        /// How much troubleshooting info to print. <, 3 and above prints response codes (404, 200, etc.), 2
        /// and above prints warnings and info."
        /// Real life testing seem to tell a different story though...
        'verbosity' => 1, // -v verbosity
        'children' => 2, // -c concurrency Number of multiple requests to make
```



```

'tries' => 10, // -n requests - Number of requests to perform
'timeout' => 0, // -t timelimit - Seconds to max. wait for responses
'auth' => false,
'proxy' => false,
'proxyauth' => false,
'target' => "",
'keepalive' => false,
'head' => false,
'interface' => "",
'respencoding' => false,
'httpversion' => CURL_HTTP_VERSION_NONE,
'cookies' => array(),
'skippercentiles' => false,
'extraheaders' => array(),

// 'internal' options
'childnr' => false,
'parentid' => false,
// the actual script path (self)
'self' => __FILE__,
'php' => 'php',
'outputformat' => 'text',
'haltonerrors' => true,
'command' => 'runparent' // allowed: 'helpmsg', 'versionmsg', 'runparent', 'runchild'
);

```

```

class ABRunner
{
    static $version = '0.1-dev';
    static $defaults = array(
        // 'real' options
        'label' => "",
        'server' => 'http://localhost', // Server hostname (the prefix for urls below).
        'urls' => 'index.php', // List of urls to test. Use double quotes around, separate them with spaces
        'urlsfile' => "",
        'repetitions' => 1000, // The number of times each client requests each url
        'concurrentcies' => '100 10', // List of concurrent clients to use
        'dognuplot' => false,
        'doaggregategraph' => false,
        'ab' => 'ab',
        'summary_file' => 'summary.txt',
        'output_dir' => 'test_logs',
        'sleep' => 1,

        // 'internal' options
        'verbosity' => 4,
    );
}

```

```
'self' => __FILE__,
'outputformat' => 'text',
'haltonerrors' => true,
'command' => 'runtests',
'abopts' => array()
);
```

Failure to meet response time requirements.

The System cannot be evaluated without completing a quantitative process to measure the response. This shall follow with more explanation.

Failure to run using particular configurations of hardware, operating systems and external libraries.

-libraries, and proper server are a critical issue in the installation and proper functioning of the Laravel framework. The absence of key libraries, PHP 5.4, PHP composer, Apache and MySQL within a proper server database are critical to the functioning of the system. The system will simply not exhibit the functionality given by the library –without warning.

Failure to gracefully handle resource shortage.

The Laravel framework implemented with a Heroku engine has the ability allow the database manager to scale in and out as well as automatic scaling to accommodate a varying number of clients in order to avoid downtime.

Failure to make resources available when they are no longer required.

A failure in the responsiveness of the SQL database can result in a cascade of failed requests and may result in a failure of the apache software and reboot incident –this is due to No code limiting the frequency of requests by any one client once a failed request has been observed.

Failure to fully recover from its own failure state or that of a related system.

And inappropriate SQL query can result into a failure of apache’s ability to respond to incoming requests which could overload incident of the system.

A failure in the PHP Laravel or composer plugins can result in a failure to interpret the code other than a string.

Testing with ApacheBenchmark:

Single instances of a client requesting a specific **number –n of times**, of which **–c are concurrent**.

In the following tables, there are 4 different instances of these simple stress tests:

First the concurrency level was changed:

Server Hostname		Document Length (bytes)	Server Software	
schedule-heroku.herokuapp.com		2 036	Apache	
Concurrency level	Time for tests (ms)	Complete Requests		Failed Requests
100	15 807	500		0
Total transferred (bytes)		HTML transferred (bytes)	Transfer rate (Kbytes/sec)	
1 104 000		1 018 000	68.21	
Connection Times (ms)				
	min	mean	median	max
Connect	22	31	30	815
Processing	60	2663	2937	3010
waiting	54	1479	1473	2999
total	92	2694	2967	3784
Time to a percent of completion (ms)				
50		75		90
2 967		3 008		3 034

Server Hostname		Document Length (bytes)	Server Software	
schedule-heroku.herokuapp.com		2 036	Apache	
Concurrency level	Time for tests (ms)	Complete Requests		Failed Requests
500	17 995	500		0
Total transferred (bytes)		HTML transferred (bytes)	Transfer rate (Kbytes/sec)	
1 104 000		1 018 000	59.91	
Connection Times (ms)				
	min	mean	median	max
Connect	24	36	30	3030
Processing	50	10 382	10 412	17 918
waiting	50	10 374	10 404	17 915
total	80	10 418	10 444	17 949
Time to a percent of completion (ms)				
50		75		90
10 444		14 218		16 511

And Secondly the Complete requests was changed more drastically:

Server Hostname		Document Length (bytes)	Server Software	
schedule-heroku.herokuapp.com		2 036	Apache	
Concurrency level	Time for tests (ms)	Complete Requests	Failed Requests	
20 000	303 427	100 000	0	
Total transferred (bytes)		HTML transferred (bytes)	Transfer rate (Kbytes/sec)	
220 800 000		203 600 000	68.06	
Connection Times (ms)				
	min	mean	median	max
Connect	20	32	30	3 036
Processing	4 202	570 449	626 140	663 901
waiting	1 319	317 062	317 014	651 050
total	4 234	570 481	626 170	663 931
Time to a percent of completion (ms)				
50	75		90	
626 170	635 745		657 250	

Reduced request by a factor of 10.

Server Hostname		Document Length (bytes)	Server Software	
schedule-heroku.herokuapp.com		2 036	Apache	
Concurrency level	Time for tests (ms)	Complete Requests	Failed Requests	
10 000	313 216	10 000	0	
Total transferred (bytes)		HTML transferred (bytes)	Transfer rate (Kbytes/sec)	
2 204 000		2 036 000	68.84	
Connection Times (ms)				
	min	mean	median	max
Connect	20	31	30	3 032
Processing	1 1142	159 379	161 661	312 704
waiting	355	158 748	161 003	312 347
total	1 174	159 410	161 689	312 738
Time to a percent of completion (ms)				
50		75	90	
161 689		237 593	312 738	

These tables only represent one instance of a client requestings –they would be variable depending on the server current capacity and load. These tests give very little indication as to how the system could be pushed past it's limits.

In order to attempt this true stress testing, the PHP classes that multiply the number of imaginary clients and their requests were used to get and average responsiveness. The variables, -n, -r and -c were set in order to recreate 1000 students 10 or 1 at a time, each with 100 requests that are made 1 at a time.

It Should be noted that the server host was changed to http://localhost:8000 –which reduced the connect times to virtually 0ms.

---

### Start Time: Wed, 06 Apr 2016 10:55:38 +0200

Testing http://localhost:8000/index.php, concurrency: 100, iterations: 100000  
Command: ab -n 100000 -c 100 "http://localhost:8000/index.php"  
This is ABRunner, Version 0.1-dev

### Start Time: Wed, 06 Apr 2016 12:36:22 +0200

Testing http://localhost:8000/index.php, concurrency: 100, iterations: 100000  
Command: ab -n 100000 -c 100 "http://localhost:8000/index.php"  
This is ABRunner, Version 0.1-dev

### Start Time: Wed, 06 Apr 2016 12:37:45 +0200

Testing http://localhost:8000/index.php, concurrency: 100, iterations: 100000  
Command: ab -n 100000 -c 100 "http://localhost:8000/index.php"

Testing http://localhost:8000/index.php, concurrency: 10, iterations: 10000  
Command: ab -n 10000 -c 10 "http://localhost:8000/index.php"  
This is ABRunner, Version 0.1-dev

### Start Time: Wed, 06 Apr 2016 12:44:17 +0200

Testing http://localhost:8000/index.php, concurrency: 100, iterations: 100000  
Command: ab -n 100000 -c 100 "http://localhost:8000/index.php"  
This is ABRunner, Version 0.1-dev

### Start Time: Wed, 06 Apr 2016 12:47:04 +0200

Testing http://localhost:8000/index.php, concurrency: 1, iterations: 1000  
Command: ab -n 1000 -c 1 "http://localhost:8000/index.php"

Testing http://localhost:8000/index.php, concurrency: 10, iterations: 10000

### End Time: Wed, 06 Apr 2016 13:02:12 +0200

---

This is a total time of 2: 07: 26 for 3.2 Million requests: an average Of 419 requests every second. For the purpose of this large test the more detailed time keeping was turned off in order to save resources.

A shorter test was then used with no concurrency –this results in fewer requests per second, however a much smaller time to do each request.

---

--

### Start Time: Wed, 06 Apr 2016 13:22:37 +0200

Testing http://localhost:8000/index.php, concurrency: 1, iterations: 100

Command: ab -n 100 -c 1 "http://localhost:8000/index.php"

Requests per second: 12.03 [#/sec] (mean)

Time per request: 83.128 [ms] (mean)

Failed requests: 0

Testing http://localhost:8000/index.php, concurrency: 10, iterations: 1000

Command: ab -n 1000 -c 10 "http://localhost:8000/index.php"

### End Time: Wed, 06 Apr 2016 13:24:13 +0200

---

--

These tests give a brief view into the resource done by the server in order to give the best responsiveness in each situation.

#### **2.2.4 Security Testing**

SQL and HTML injection can be used to verify the security of a system and its capacity to leak important system information about the structure of the object oriented HTML/PHP or of the database schema; such as the exact HTML/PHP structure, Files, Extensions, Updates or even Database structure and contents. This is clearly an outcome to be avoided: thus a Laravel PHP framework was implemented in order to prevent eventual security breaches –especially by injection. The higher level abstraction of the frameworks helps to achieve this.

If this nikto is directly implemented into the server location, then many information is readily available:

```

Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>perl nikto.pl -h https://schedule-heroku.herokuapp.com/
Can't open perl script "nikto.pl": No such file or directory

C:\WINDOWS\system32>cd\Users\Nicolas\Documents\GitHub\nikto\program

C:\Users\Nicolas\Documents\GitHub\nikto\program>perl nikto.pl -h https://schedule-heroku.herokuapp.com/
- Nikto v2.1.6
-----
+ Target IP:          23.23.231.101
+ Target Hostname:    schedule-heroku.herokuapp.com
+ Target Port:       443
-----
+ SSL Info:          Subject: /C=US/ST=California/L=San Francisco/O=Heroku, Inc./CN=*.herokuapp.com
+                   Ciphers: ECDHE-RSA-AES128-GCM-SHA256
+                   Issuer: /C=US/O=DigiCert Inc/OU=www.digicert.com/CN=DigiCert SHA2 High Assurance Server CA
+ Start Time:       2016-04-05 12:46:54 (GMT-4)
-----
+ Server: Apache
+ Retrieved via header: 1.1 vegur
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The site uses SSL and the Strict-Transport-Security HTTP header is not defined
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ Server banner has changed from 'Apache' to 'Cowboy' which may suggest a WAF. Load balancer or proxy is in place
+ Server leaks inodes via ETags, header found with file /robots.txt, fields: 0x18 0x52fb5ba3254c0
+ Server is using a wildcard certificate: *.herokuapp.com
+ Allowed HTTP Methods: GET, HEAD
+ OSVDB-3092: /web.config: ASP config file is accessible.

```

However, if the system is properly uploaded and the nikto is ran from outside the system, then no web server is detected at all.

## 3. System Delivery

### 3.1 Installation Manual

This installation manual will explain to an administrator how to install the scheduler system on a local hosting server. It is also possible to do it on a shared server, meaning the website would be accessible via the internet. However, this requires two things: a domain name and a shared server capable of running Laravel, such as <https://www.fortrabbbit.com/>. Both cost a certain amount of money. However, the installation manual will only show how to install the project locally on the computer. The following installation is performed on Windows 10, but it is also possible to do it on other OS, with slightly modifications.

4 software are mainly required, with the scheduler project:

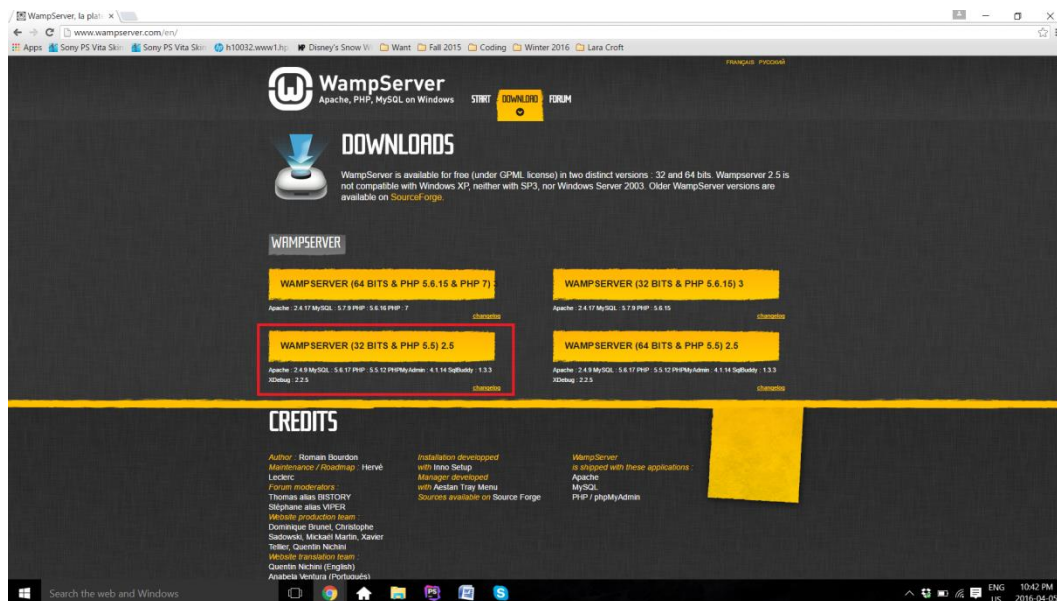
- WampServer
- Visual Studio

- Composer
- MySQL
- Scheduler Project from GitHub

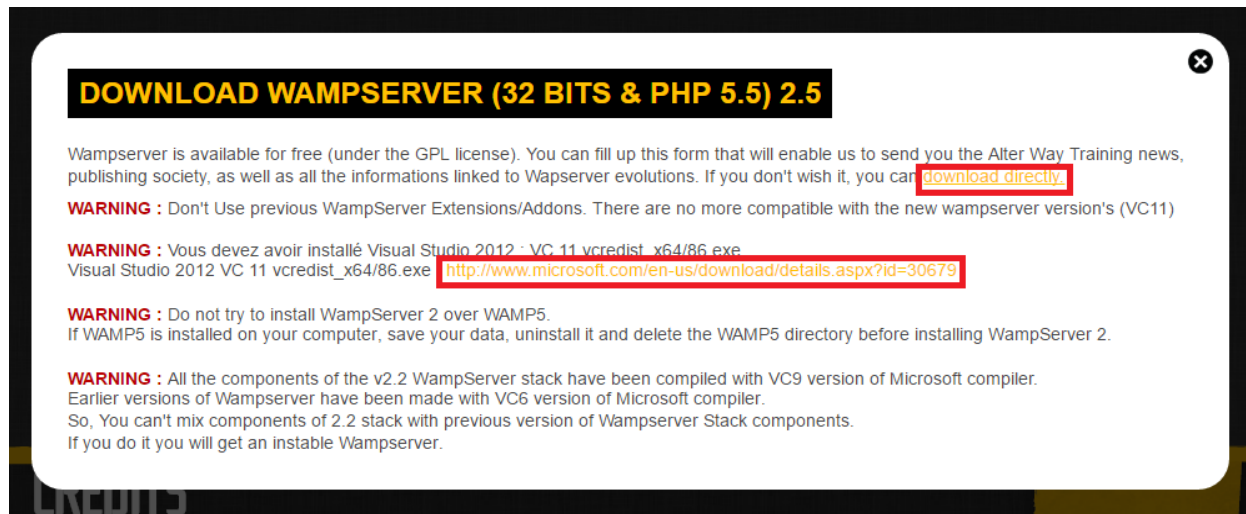
## **Step 1: WampServer Installation**

The first step consist of downloading and installing WampServer. WampServer contains all the needed softwares to run a PHP program on the local host: PHP, Apache and MySQL. It is recommended to download WampServer 2.5, containing especially Apache 2.4.9, MySQL 5.6.17, PHP 5.5.12. The scheduler has been coded in PHP 5.5, therefore it is preferred to use that version in case syntax modifications have been done in the newer versions of PHP 7. Furthermore, this package works both on 32-bit and 64-bit computers. The website to this download is <http://www.wampserver.com/en/>.

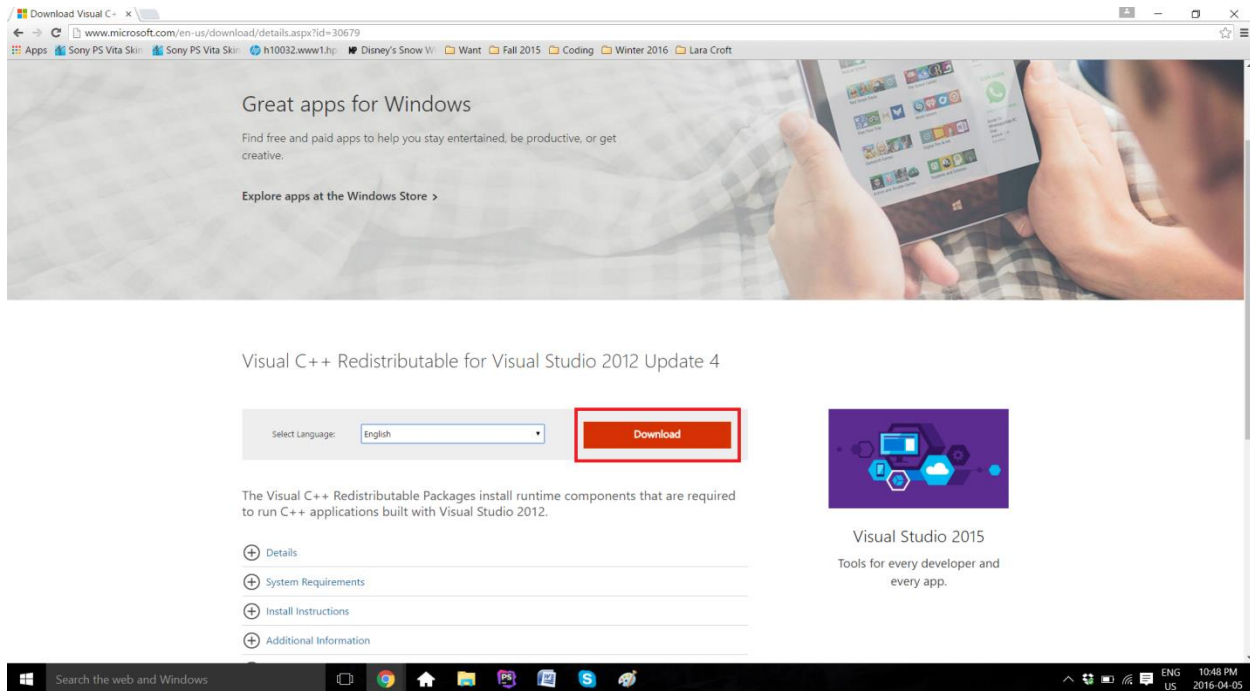
**Note:** Another alternative to WampServer is XAMPP. The only difference between both is that XAMPP is offered on multiple operating systems, such as iOS and Linux, in addition to Windows. WampServer is only offered for Windows.



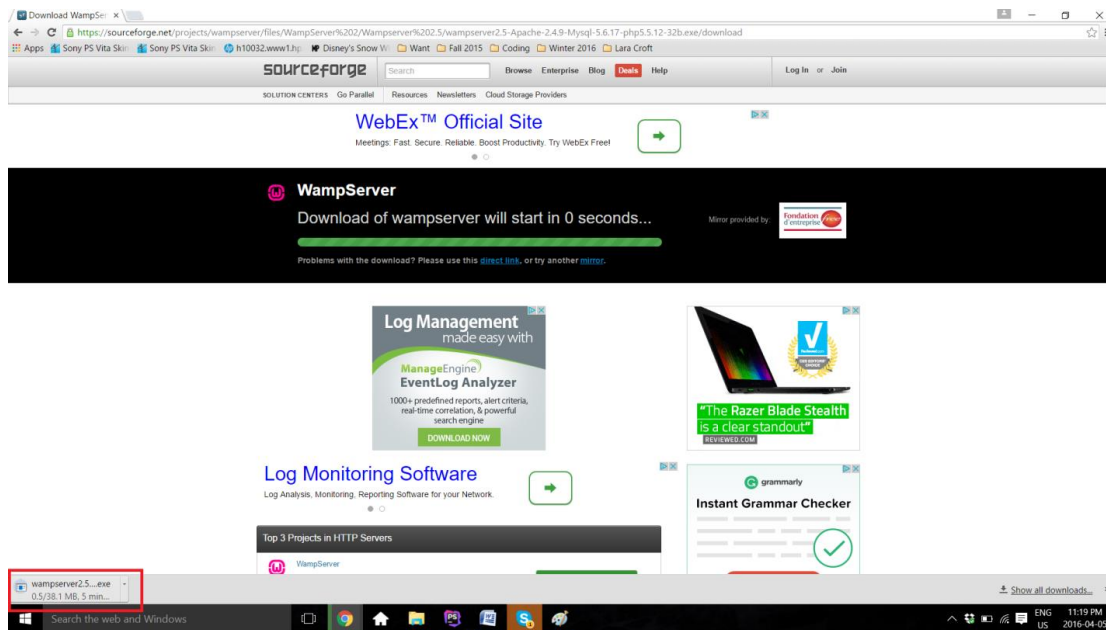




As shown here, there is a couple of warnings regarding WampServer. It is especially important to pay attention to second **warning**. In order for WampServer to work, Visual Studio is required. This is because Apache works along with Visual Studio. If Visual Studio 2012 is not installed on the computer, the administrator should click on the link <https://www.microsoft.com/en-us/download/details.aspx?id=30679> found in the same warning. This opens a Microsoft download page for the correct version of Visual Studio. Once downloaded, Visual Studio can be installed. No special modification is required, therefore it is only necessary to follow the instructions displayed on Visual Studio installation window and click on **Next** a couple of times until the installation has begun, and proceeded successfully. It is important to note that there is a paid version of Visual Studio, but it is not required as the free version works equally well for the purpose of Apache.



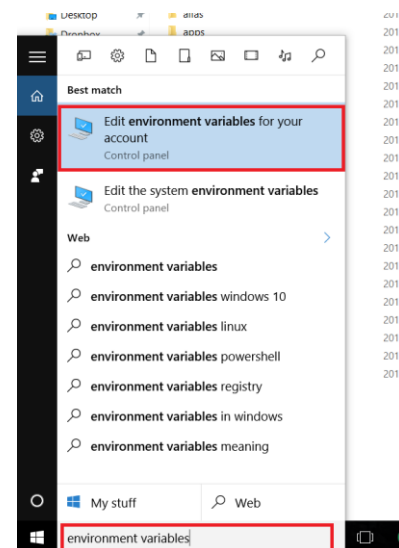
Once Visual Studio is set, the administrator can now go back to the WampServer website and click **download directly**. A new page from the website SourceForge will open, containing the file. A few seconds has to pass for the download to start. On Google Chrome, the download will be shown on the bottom left of the web page. After the download is completed, it is only sufficient to click on the file and the installation will begin. Otherwise, as any other downloads, the file has to be located to wherever downloaded files are usually saved (it is usually in the **Download** directory).

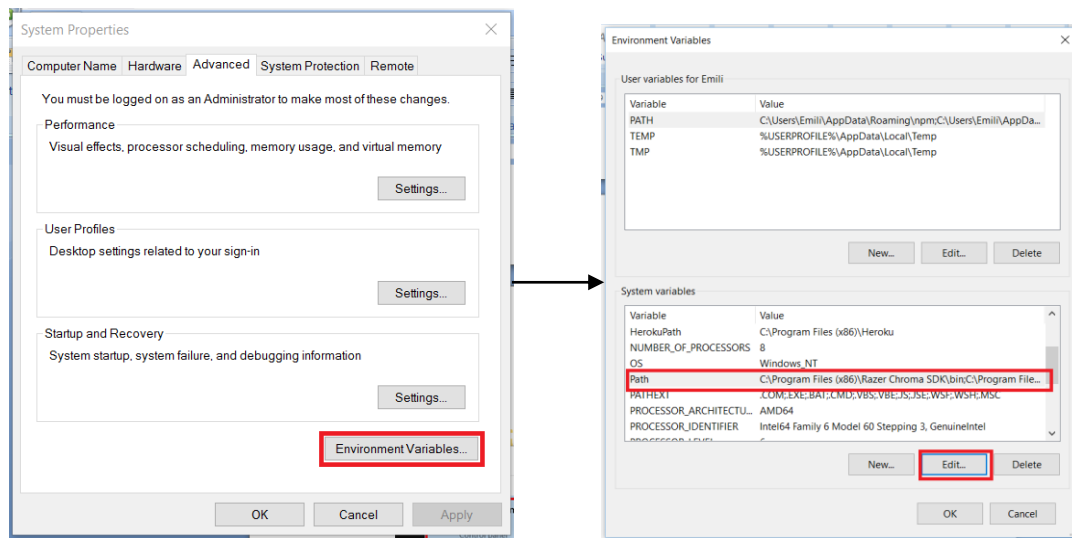


Once again, installing WampServer does not require any particular specifications or changes on the automatic procedure. The only thing required from the administrator is to keep track of where the software is installed. In this example, the software has been saved under **C:\wamp**.

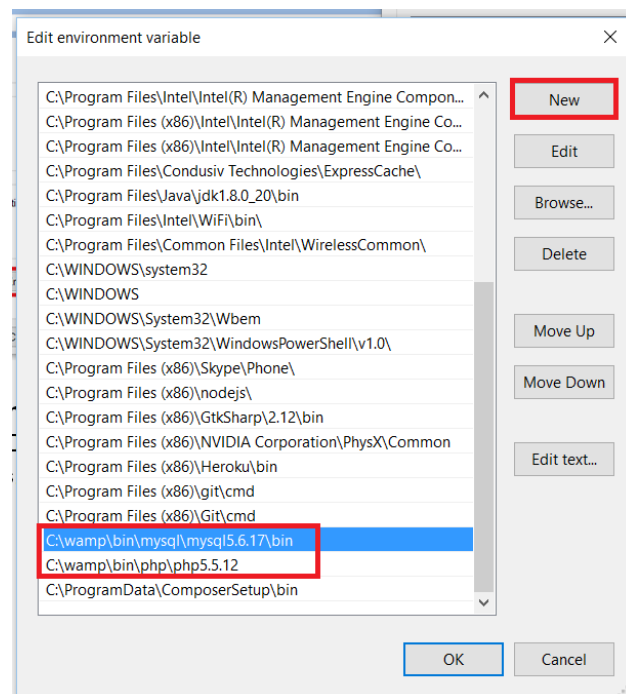
## **Step 2: Create the PHP and MySQL Path in the Environment Variables.**

This step enables the Command Prompt to have access to the PHP and MySQL, therefore being able to manipulate and launch the software from there later on in the procedure. First, it is possible to find the Environment Variables by typing the name in the **Search** available on the computer.





The control panel of the **System Properties** opens, under the **Advanced** section. **Environment Variables..** can be found there. Once clicked, another window called **Environment Variables** will open. In the **System variables**, **Path** has to be located, and then edited. Two new variables have to be added. The first one is the path to the Wamp's MySQL bin file. The file has to be located into the computer, as it is **C:\wamp\bin\mysql\mysql5.6.17\bin** in this case. The second one is the path to the Wamp's PHP file, as it is **C:\wamp\bin\php\php5.5.12** in this case. Once everything is set, the administrator has to click **OK** on all three windows, therefore saving the changes and closing the **Control Panel**.



To make sure the path worked, the following commands have to be typed on the Command Prompt and the display should be similar to the following image, without any error message:

- **php -v**
- **mysql -u root -p**
  - This one will ask for a password, the password is an empty string, therefore it is only necessary to click on **Enter** on the keyboard.

```
Command Prompt - mysql -u root -p
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\Emili>php -v
PHP 5.5.12 (cli) (built: Apr 30 2014 11:20:55)
Copyright (c) 1997-2014 The PHP Group
Zend Engine v2.5.0, Copyright (c) 1998-2014 Zend Technologies
    with Xdebug v2.2.5, Copyright (c) 2002-2014, by Derick Rethans

C:\Users\Emili>mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 22
Server version: 5.6.17 MySQL Community Server (GPL)

Copyright (c) 2000, 2014, Oracle and/or its affiliates. All rights reserved.

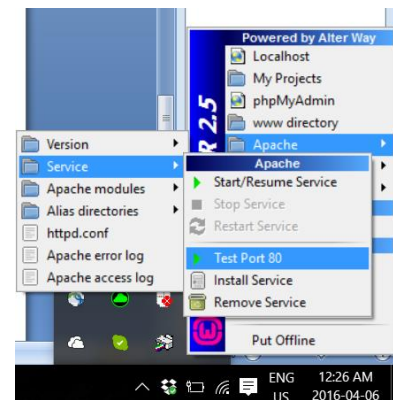
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

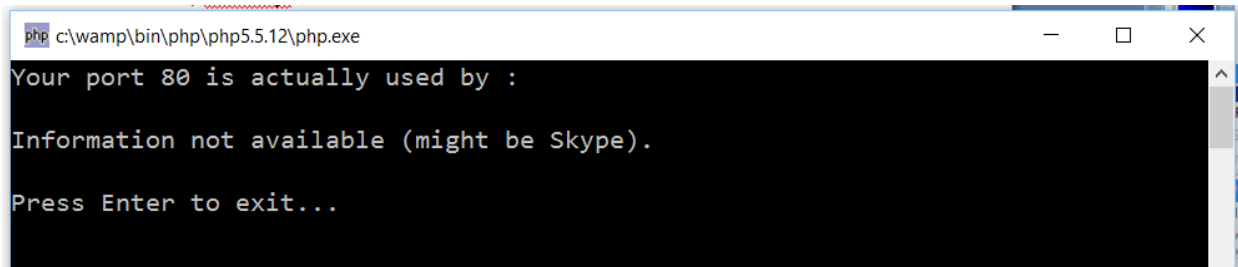
mysql>
```

### Step 3: Turn on WampServer and Test the Connection

First, Wamp has to be turned on. If the color does not turn green after clicking **Start All Services** and the color is stuck at orange, it is most likely because **port 80**. WampServer, as well for XAMPP, uses **port 80** for Apache. Therefore, this error is probably caused because another application is already using **port 80**. It is possible to test the connection by following this path on the Wamp application: **Apache > Service > Test Port 80**.



This is going to display an error if the port is really not available for Apache. As shown here, the error is usually caused by Skype being on at the same time, as both Apache and Skype use the same port. If it is the case, Skype has to be completely closed. It is then only sufficient to **Restart All Services** on Wamp, and the icon should turn green.



```
c:\wamp\bin\php\php5.5.12\php.exe

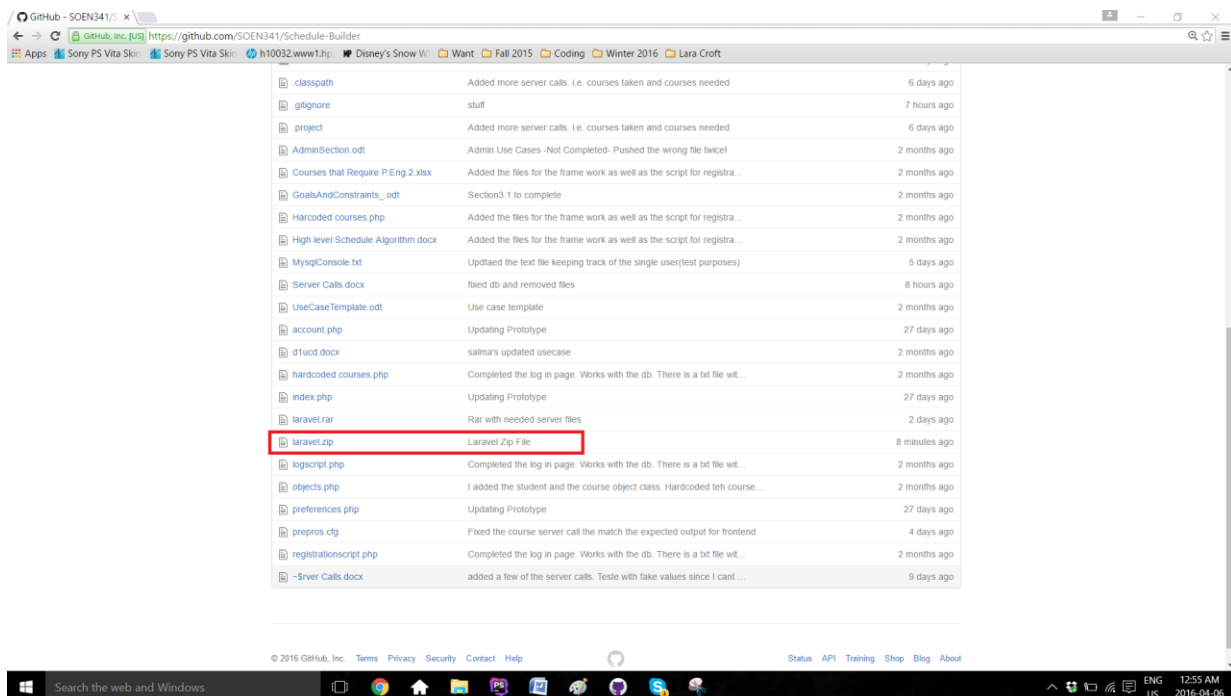
Your port 80 is actually used by :

Information not available (might be Skype).

Press Enter to exit...
```

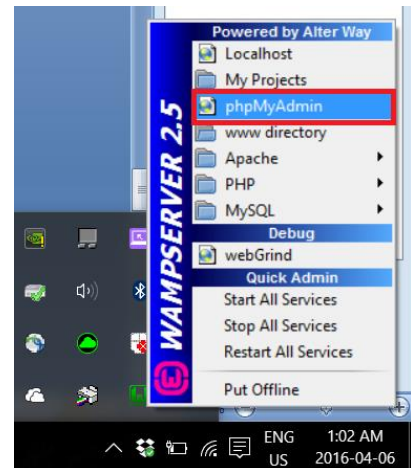
#### Step 4: Download the project from GitHub

The project has to be now retrieved from GitHub. This can be done from the following link: <https://github.com/SOEN341/Schedule-Builder>. As it is a zip file, the following has to be unzipped. The file can then be placed at any desired location.

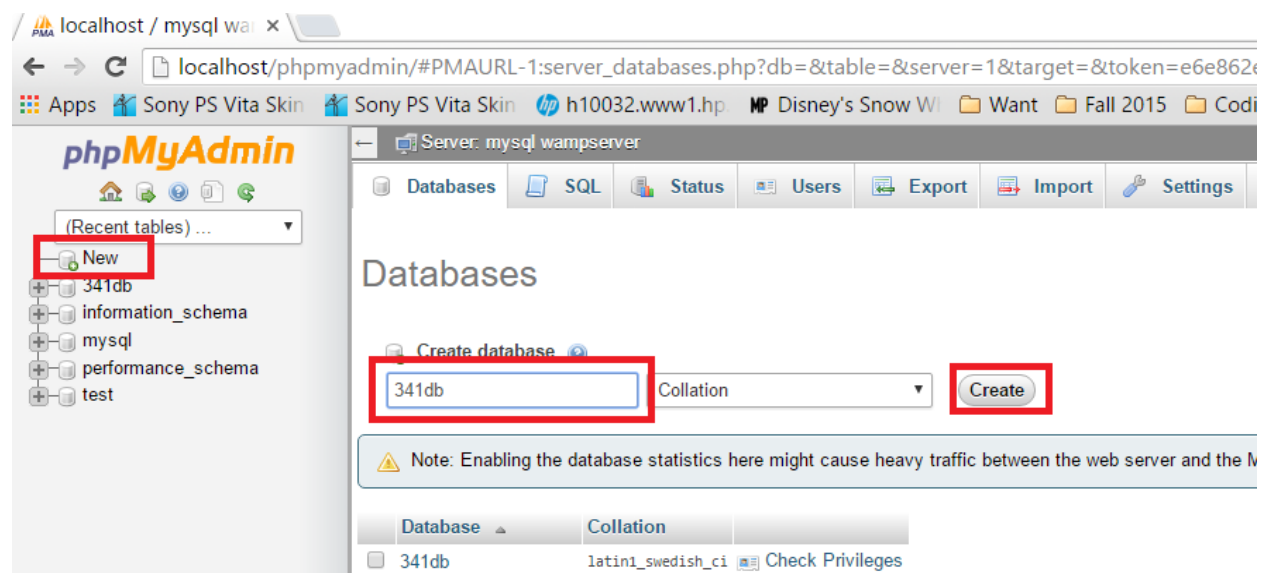


## Step 5: Set Database

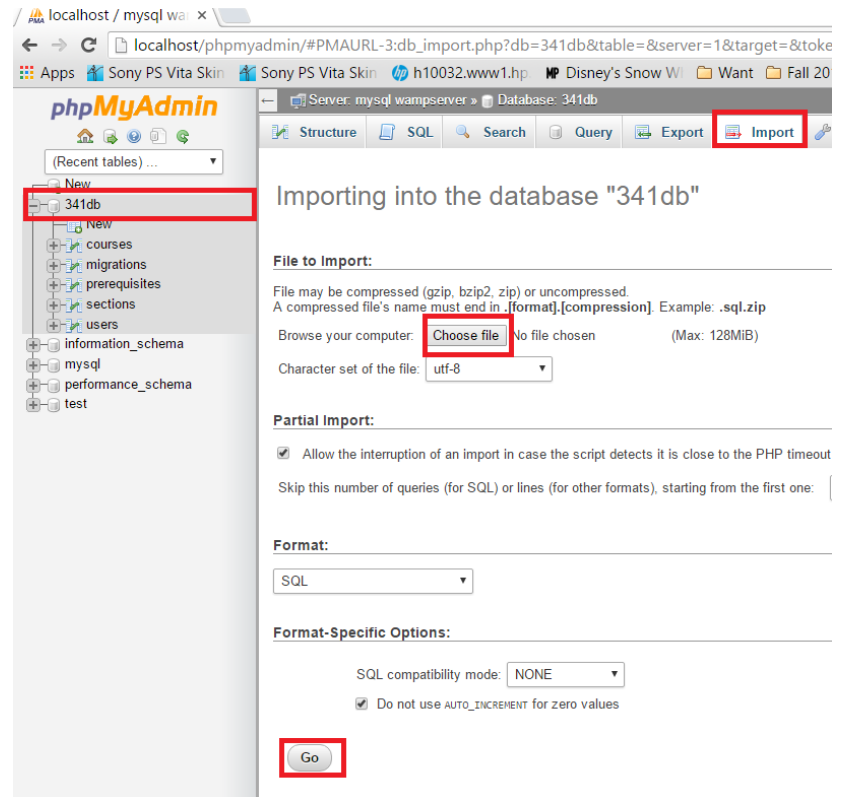
In order for the software to work, it has to be connected to the Database. From the **Laravel** file downloaded from GitHub, there is a file called **finaldbfile.sql**. This is the entire database of the project, containing all courses, sections, users information. This database has to be uploaded on MySQL on Wamp. To do this, **phpMyAdmin** has to be opened via Wamp.



**phpMyAdmin** will then open on the default browser. Then, the database has to be created, by clicking on **New**. The name of the database is **341db**.



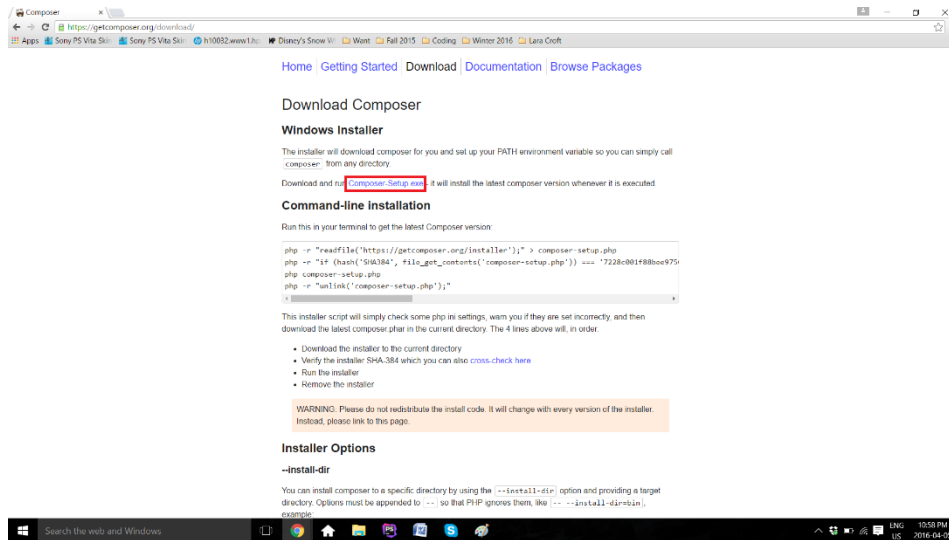
Afterwards, **finaldbfile.sql** has to be imported onto the same **341db** database. On **phpMyAdmin**, **341db** has to be selected, then **Import**, then **Choose file**. Navigate to the **Laravel** file and select **finaldbfile.sql**. Once the upload done, this section can be finalized by clicking **Go**. A green confirmation message will be displayed, saying the import has been successful.



## Step 6: Download and Install Composer

Composer is a dependencies manager for PHP. This means that Composer will handle all the needed libraries and files, in order to ease the development of the project. It can be download from <https://getcomposer.org/download/>. It is once again only necessary to follow the installation instructions. However, it is important keep track of where the Composer gets the php.exe file from. It has to be from Wamp. If it is not the case, relocate the path to the **php.exe** inside of Wamp. The path is **C:\wamp\bin\php\php5.5.12\php.exe** for this case.

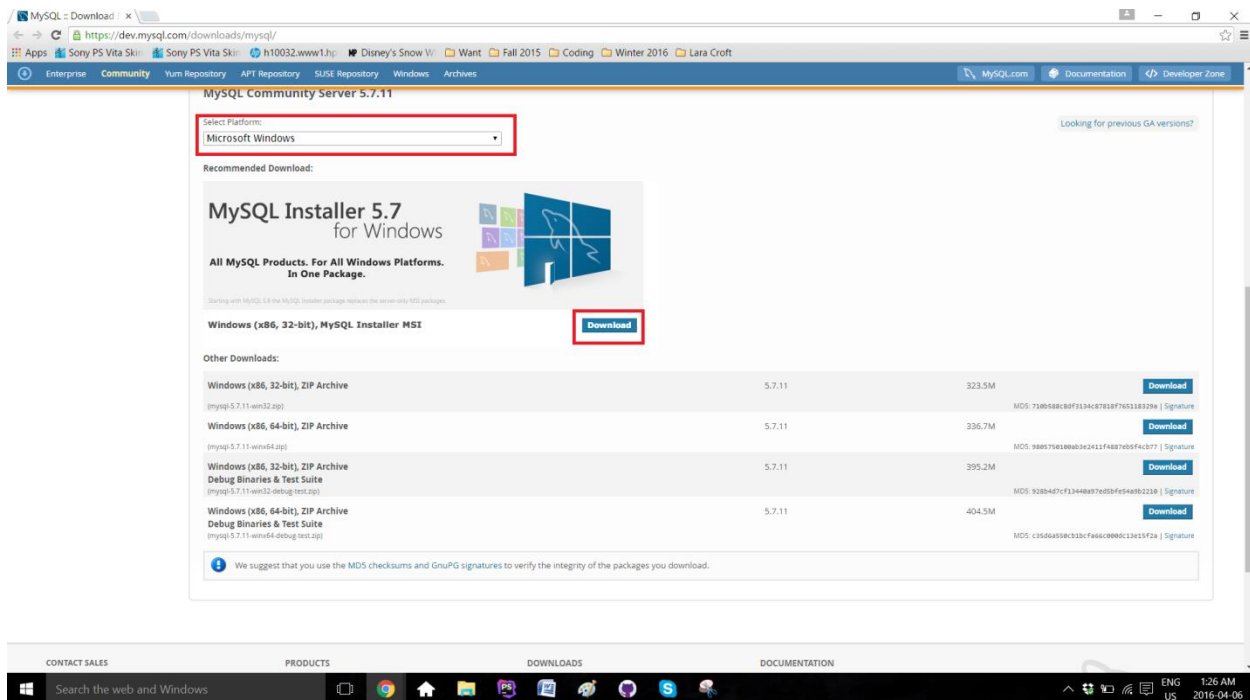




## Step 7: Download and Install MySQL

Even though MySQL has been downloaded using Wamp, there seem to be some connection problem from Wamp if the software is not directly inside its **www** directory. To solve this problem, the original MySQL software is installed again, which helps to install MySQL correctly across the whole system. This allows the project to see the available MySQL throughout the system.

The software can be downloaded from <https://dev.mysql.com/downloads/mysql/>. The download can be found under **Community Server**. The desired OS can be picked (Windows in this case). **MySQL Installer 5.7 for Windows** is the desired version.



As any other past installation, it is only necessary to follow the instructions provided by the installer. Once installed, it is recommended to test if MySQL is setup correctly, by typing **mysql --version** on **Command Prompt**.

```

C:\Users\Emili>mysql --version
mysql Ver 14.14 Distrib 5.6.17, for Win32 (x86)

```

## Step 8: Run Scheduler Project

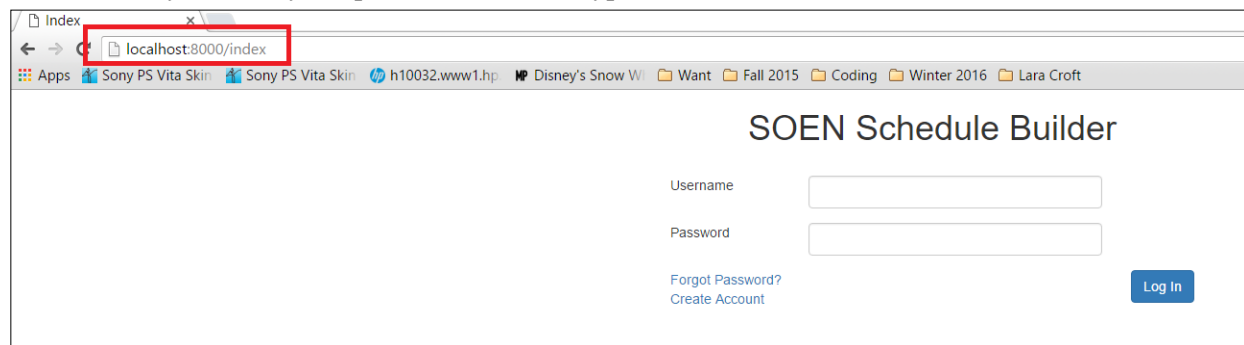
Once everything is set up and Wamp is turned on and green, the project can now run. To do so, **Command Prompt** is used. The first command to type is **cd \*project location\***. To navigate backward in folders, the command **cd..** does the trick. Once in the file of the project, the last command is executed: **php artisan serve**. The output **Laravel development server started on http://localhost:8000/** should be displayed.

```
Command Prompt - php artisan serve
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\Emili>cd Documents\GitHub\Schedule-Builder\laravel

C:\Users\Emili\Documents\GitHub\Schedule-Builder\laravel>php artisan serve
Laravel development server started on http://localhost:8000/
```

Now, it is only necessary to open a browser, and type **localhost:8000** in the URL bar.

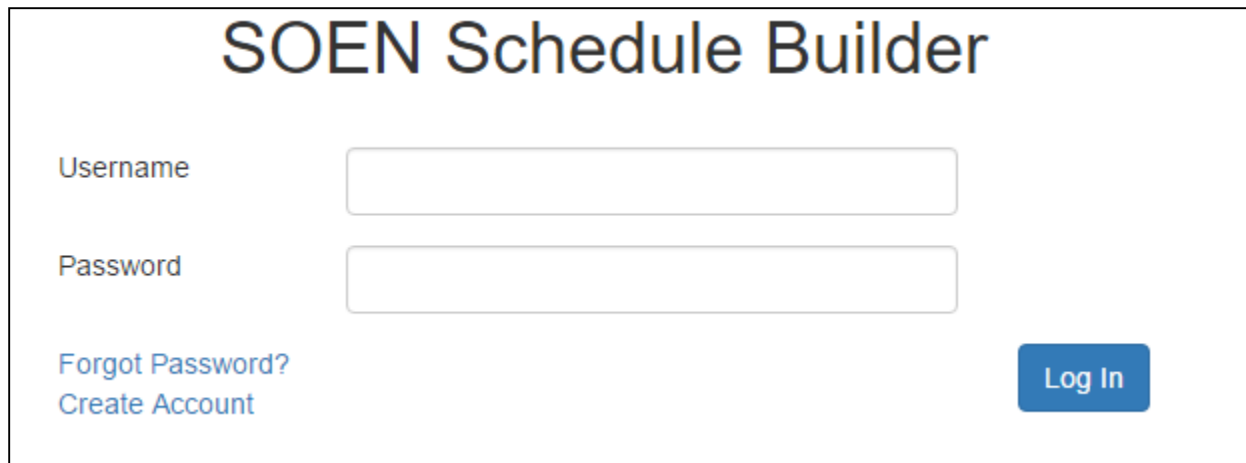


### 3.2 User Manual

The Scheduler is a system used by undergraduate software engineering students at Concordia University, which generates a class schedule based on the student's record. It can be accessed from: <https://schedule-heroku.herokuapp.com/index>.

#### Logging In

Upon entering the website, a username and password will be needed to identify the user and allow access to the Scheduler. Returning users can simply login, while new users will be required to create a new account using the button "Create Account".

The image shows the home page of the SOEN Schedule Builder. At the top, the title "SOEN Schedule Builder" is displayed in a large, dark blue font. Below the title, there are two input fields: "Username" and "Password". To the left of the "Password" field, there are two links: "Forgot Password?" and "Create Account". To the right of these links is a blue button labeled "Log In".

# SOEN Schedule Builder

Username

Password

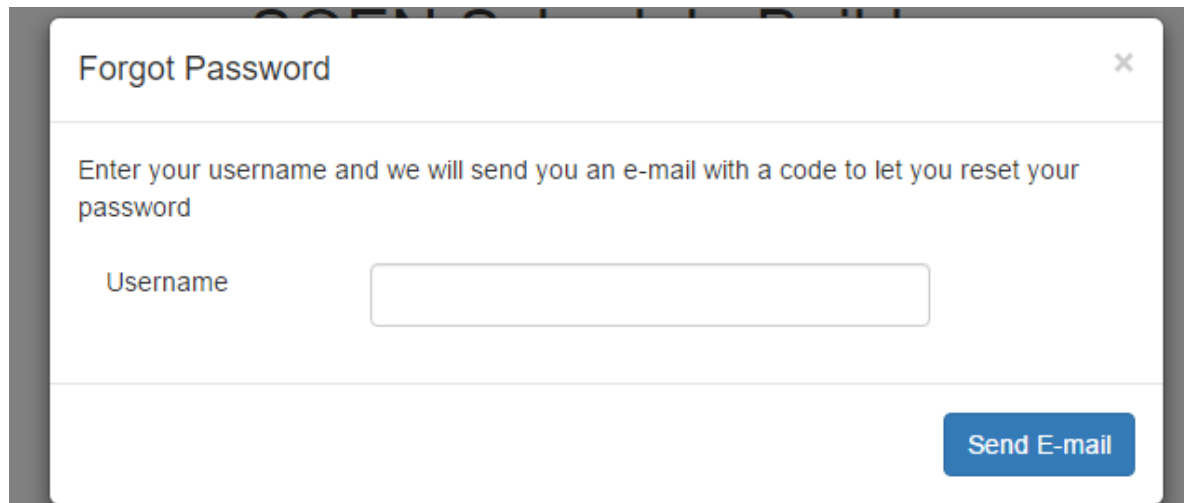
[Forgot Password?](#)  
[Create Account](#)

[Log In](#)

**Figure 1:** *Home Page*

### **Resetting the Password**

If one of the returning users forgot their password, they can press the “Forgot Password?” link, which will allow them to reset their password by receiving a temporary one through e-mail. Successfully logging in will grant access to the menu page.

The image shows a modal window titled "Forgot Password" with a close button (X) in the top right corner. Inside the modal, there is a text prompt: "Enter your username and we will send you an e-mail with a code to let you reset your password". Below this prompt is a text input field labeled "Username". At the bottom right of the modal is a blue button labeled "Send E-mail".

## Forgot Password

Enter your username and we will send you an e-mail with a code to let you reset your password

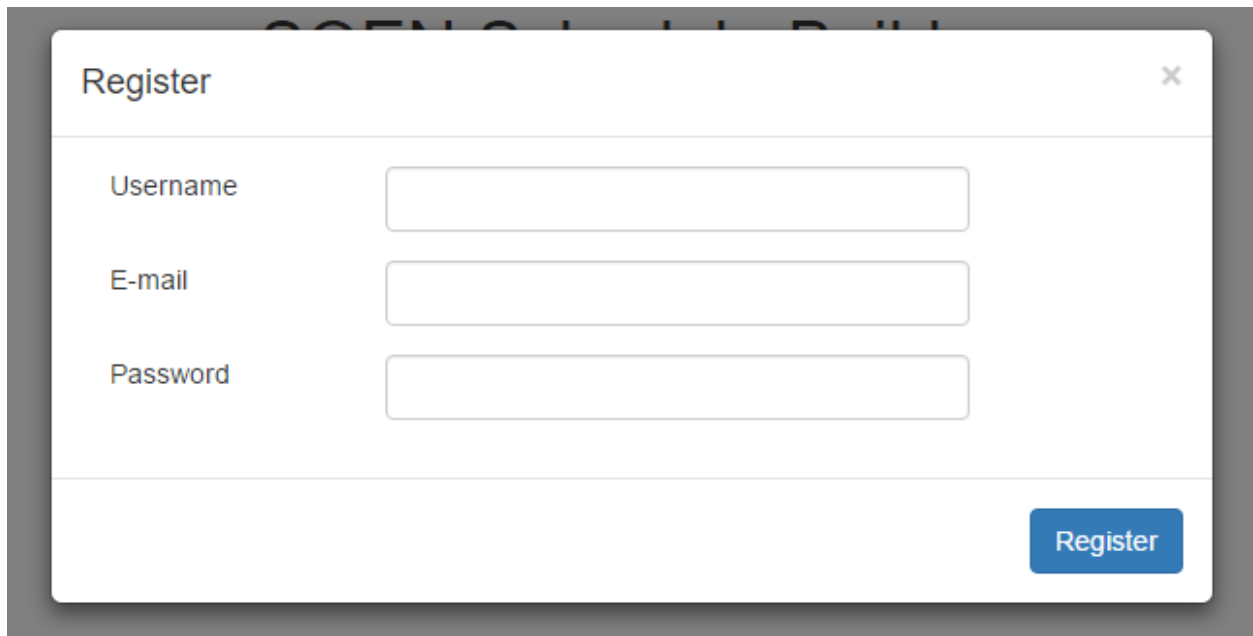
Username

[Send E-mail](#)

**Figure 2:** *Resetting Password*

### **Account Creation**

When first time users click on “register”, the screenshot below will be shown. They will simply have to follow the instruction and fill out each box with the required information. Once this is done, the menu page will open.

A screenshot of a 'Register' form. The form has a title 'Register' in the top left corner and a close button 'x' in the top right corner. It contains three input fields: 'Username', 'E-mail', and 'Password'. A blue 'Register' button is located at the bottom right of the form.

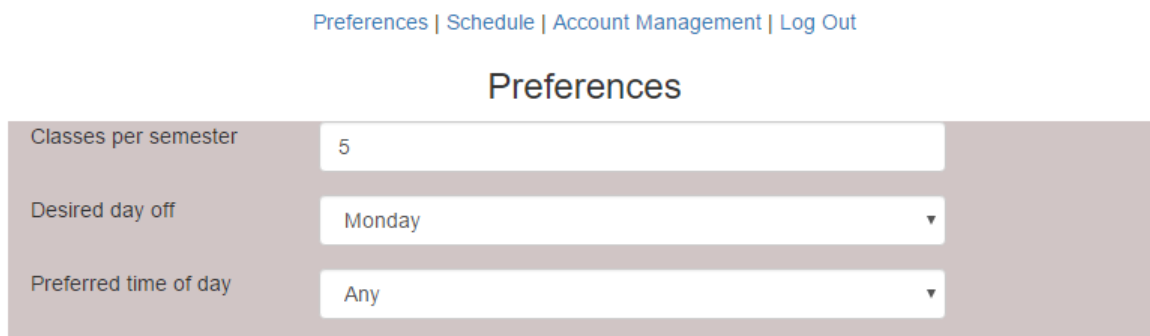
**Figure 3:** *Signing up*

The menu page is accessed after entering a valid username and password or after creating an account. This is where students make all their decisions concerning their schedule.

Here are the different actions users can choose in the menu page:

### **Preferences:**

In this page, the users will first see when and where they would like to take courses. As seen in the screenshot below, the choices of having a day of the week off, taking classes in the morning afternoon or at night and the number of classes to take in the semester are all preferences to be set.

A screenshot of the 'Preferences' page. At the top, there is a navigation bar with links: 'Preferences | Schedule | Account Management | Log Out'. Below this, the title 'Preferences' is centered. The form contains three rows: 'Classes per semester' with a text input field containing '5'; 'Desired day off' with a dropdown menu showing 'Monday'; and 'Preferred time of day' with a dropdown menu showing 'Any'.

**Figure 4:** *Modifying preferences*

### **Adding classes**

The preferences page also allows the students to input their taken courses as well as the needed courses.

One way to add the taken classes by choosing the semester already completed and generating a course list of all the courses that were offered. The user can then simply choose the courses he or she already completed.

The other method is by using the “Add Class” button and manually writing the course name number.

For the needed courses, the same concept is used. Using the “Add Class”, courses can be saved. Each of the saved courses can be edited or removed using the two icons located next to the course numbers.





Using the “build schedule” button seen at the bottom of the page, schedules will be generated and can be accessed in the “Schedule” page.

**Classes**

Semesters Taken



Generate course list

Courses Taken (showing 2 of 2) ▲

Class Name	Course Number	
Object Oriented 1	COMP 248	 
Object Oriented 2	COMP 249	 

Add Class

Courses Needed (showing 1 of 1) ▲

Class Name	Course Number	
Data Structures and Algorithms	COMP 352	 

Add Class

Build Schedule

**Figure 5:** *Adding courses*

### **Account Management:**

By clicking on the account information link, users can modify their username, email or password (see figure 6).

## Account Information

Username: **soen341**

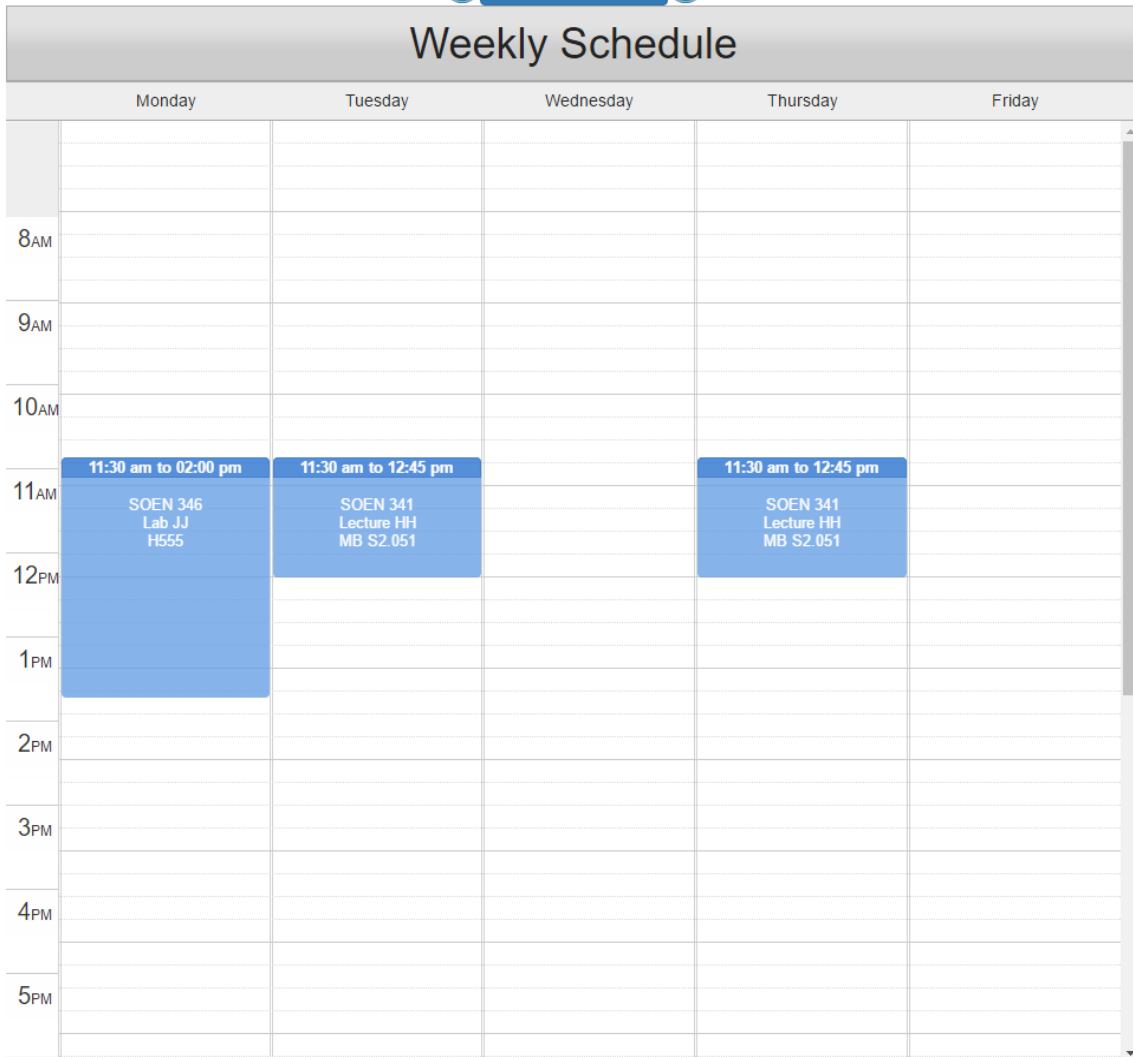
E-mail: **soen341@xx.com**

**Figure 6:** Account information

### **Schedule:**

The schedule page is where all the possible schedules corresponding to user's preferences are going to be generated. It is to be noted that the Scheduler needs the user to input all their taken and needed classes in order to generate a schedule.

The various schedules can be navigated by using the arrows at the top. When a schedule is deemed acceptable, it can be selected using the "Select This Schedule" button. Upon selecting a schedule, the sequence for the rest of the academic years will be created giving a guideline to follow (see figures below).



**Figure 7:** *Schedule*



Year 1 Fall	
Class Number	Course Name
COMP 249	Object Oriented Programming 2

Year 1 Winter	
Class Number	Course Name
COMP 249	Object Oriented Programming 2
COMP 249	Object Oriented Programming 2

Year 2 Fall	
Class Number	Course Name
COMP 249	Object Oriented Programming 2

Year 2 Winter	
Class Number	Course Name
COMP 249	Object Oriented Programming 2
COMP 249	Object Oriented Programming 2

Year 3 Fall	
Class Number	Course Name
COMP 248	Object Oriented Programming 1

Year 3 Winter	
Class Number	Course Name
COMP 249	Object Oriented Programming 2
COMP 249	Object Oriented Programming 2

**Figure 8:** *Full sequence*

**\*Disclaimer:** The current state of the system is not completely functional and once all the issues are corrected, a full schedule will be displayed (which will be the case for the last deliverable). The screenshots are also going to be modified once the system is functioning properly.

### **Logging out**

To log out, the “log out button” has to be pressed. This will exit the system bring the user back to the home page.

### **3.3 Admin Manual**

The Admins access the website from the same domain name: <https://schedule-heroku.herokuapp.com/index>.

### **Logging Page**

Upon entering the website, a username and password will be demanded. Admins should have a special login username and password to enter the website specifically designed for them.

If the password is forgotten, by clicking on “Forgot Password”, it can easily be recovered.

Successfully logging in brings the Admins to the main page of the website.

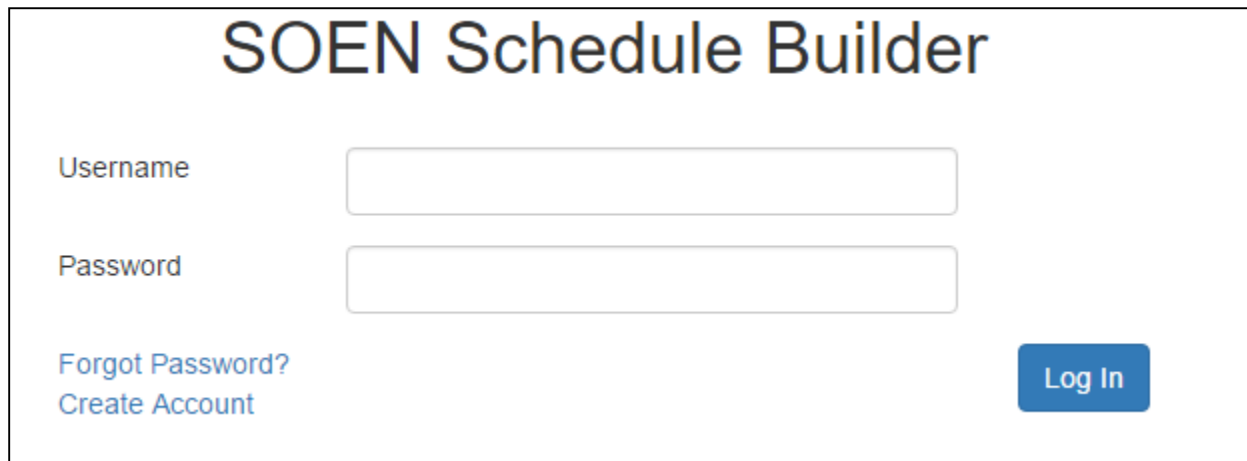
The image shows a login page for 'SOEN Schedule Builder'. At the top, the title 'SOEN Schedule Builder' is displayed in a large, bold, black font. Below the title, there are two input fields: 'Username' and 'Password'. The 'Username' field is a simple white box with a light gray border. The 'Password' field is a white box with a light gray border and a small eye icon on the right side to toggle visibility. Below the 'Password' field, there are two links: 'Forgot Password?' and 'Create Account', both in a blue font. To the right of these links is a blue button with the text 'Log In' in white.

Figure 9: Home Page

The Admins can add or edit courses present in the database. Entering this page provides the Admins with a whole list of the courses.

### **Adding a Course**

In order to add a course, the button “add class” is to be pressed. This will provide the Admins with a form to fill. More specifically, entering the course number, name and the semester.

### **Adding/Modifying Information and Sections**

Next to courses seen on the page, there is a small icon showing a paper and a pencil. By pressing it, it allows the admins to change the descriptions of the course or its number. Also, by clicking on the course name, the different sections of the course are displayed. The same paper and pencil icon can be pressed to modify the time of the sections and the classroom.

[Log Out](#)



Class Name	Course Number	Semester	Description	Credits	
<a href="#">Object Oriented Programming 1</a>	COMP 248	Fall	Introduction to programming. Basic data types, variables, expressions, assignments, control flow. Classes, objects, methods.	3	
<a href="#">Object Oriented Programming 2</a>	COMP 249	Winter	Introduction to programming. Basic data types, variables, expressions, assignments, control flow. Classes, objects, methods.	3	

Figure 10: Course list

[Log Out](#)


Section	Course Number	Type	Day	BeginTime	EndTime	Classroom	
JJ		Lecture	1	11:30	14:00	H555	
HH		Lecture	13	11:30	14:00	H321	

Figure 11: Class Sections

## Log Out

To log out, Admins can press the “log out” button at the top of the page.

## 4. Final Cost Estimate

A number of design and testing aspects were overlooked during our initial estimate. In the first deliverable, tasks such as creating the database, and ensuring that the system would be able to perform the necessary read and write operations were slightly underestimated. Testing was also

The total project cost estimate as calculated to date is detailed in the table below. A side by side comparison between the original estimated cost and the actual cost illustrates the difference in hours that our team need to make adjustments for.

Artifact		Estimated Cost in Hours	Final Cost in Hours
Deliverable0		5	5 (\$125)
Deliverable1		70	70 (\$1750)
Deliverable2			
4+1 Architectural View	Logical	10	10
	Development		3
	Physical		3
	Process		5
	Scenarios		1

<b>Subsystem Interface Specification</b>	<b>25</b>	<b>20</b>
<b>UML Class Diagram</b>	<b>12</b>	<b>10</b>
<b>Dynamic Design Scenario</b>	<b>6</b>	<b>10</b>
<b>Estimation</b>	<b>7</b>	<b>5</b>
<b>Rapid Prototyping Report</b>	<b>22</b>	<b>25</b>
<b>Testing</b>	<b>15</b>	<b>15</b>
<b>Risks</b>	<b>4</b>	<b>4</b>
<b>Total Hours</b>	<b>101</b>	<b>111</b>
<b>Cost Estimate (\$25/hr)</b>	<b>\$2525</b>	<b>\$2775</b>
<b>Deliverable3</b>		
<b>Database Creation</b>	<b>20</b>	<b>20</b>
<b>Web Interface Design</b>	<b>20</b>	<b>16</b>
<b>Database interaction &amp; Server calls</b>	<b>20</b>	<b>14</b>
<b>Unit Testing</b>	<b>20</b>	<b>30</b>
<b>Requirement Testing</b>	<b>20</b>	<b>20</b>
<b>Stress Testing</b>	<b>8</b>	<b>10</b>
<b>Security Testing</b>	<b>7</b>	<b>7</b>
<b>Installation</b>	<b>Unaccounted for</b>	<b>10</b>
<b>Installation Manual</b>	<b>6</b>	<b>12</b>
<b>User's Manual</b>	<b>6</b>	<b>8</b>
<b>Administrator manual</b>	<b>Unaccounted for</b>	<b>2</b>
<b>Estimation</b>	<b>4</b>	<b>3</b>
<b>Total Hours</b>	<b>129</b>	<b>152</b>
<b>Cost Estimate (\$25/hr)</b>	<b>\$3225</b>	<b>\$3800</b>

An additional cost of 23 hours has been incurred during the realization of the third deliverable bumping the cost from an estimated \$3225 to an actual \$3800. A 50hours estimate is still projected for the realization of the final deliverable as per our initial calculations mainly due to the adjustments that are to be made in order to correct t and adjust the aspects of the project that were deemed unsatisfactory.

It is therefore our estimation that the total cost of the project will be increased by \$575 which corresponds to the difference between the estimated and actual cost incurred for the realization of deliverable3.

The final cost jumps from the \$9050 estimated at the end of deliverable2 to \$9625.