

[**Directed Independent Study**](http://digitalcommons.unf.edu)

[UNF School of Computing](http://digitalcommons.unf.edu/etd) [Fall 2019](http://digitalcommons.unf.edu/student_scholars)

Leadership Hours Tracking

© 2019 All Rights Reserved

LEADERSHIP HOURS TRACKING

School of Computing

Guidance:

Professor Sherif Elfayoumy

Professor Larry Snedden

Professor Walt

UNIVERSITY OF NORTH FLORIDA

SCHOOL OF COMPUTING

December 2019

ii

CONTENTS

|  |  |
| --- | --- |
| Chapter 1. Introduction ............................................................................................................. | 1 |
| Chapter 2. Hardware & Software Specification…................................................................... | 2 |
| Chapter 3. Database Design………….……….…………......................................................... | 3 |
| Chapter 4. Architecture Pattern………….……………............................................................ | 8 |
| Chapter 5. ADO.NET Entity Framework…………….............................................................. | 11 |
| Chapter 6. Design……………………….………………......................................................... | 17 |
| Chapter 7. Data Export and Import……….……………............................................................ | 23 |
| Chapter 8. Security……….…………….................................................................................... | 24 |
| Chapter 9. IIS – Internet Information Services…………........................................................... | 25 |
| Chapter 10. Walkthru…………………….……………............................................................. | 27 |

Appendix A: Deliverable 37

Appendix B: Database Schema 38

References 40

Chapter 1

INTRODUCTION

At present, the hours spent by students participating in various events and activities are manually maintained by various faculties. It has been time consuming task and difficult to access and compile information and to report on timely fashion. “Leadership hours Tracking” is a web based application developed to automate the data capture and reporting.

Chapter 2

HARDWARE AND SOFTWARE REQUIREMENTS

The hardware and software specifications are listed below. “Leadership Hours Tracking” application is developed and tested on this platform.

Hardware:

Windows 2000 Virtual Machine: CICSVM-SOCHONORS

CPU:

Memory:

Software:

|  |  |  |  |
| --- | --- | --- | --- |
| **Serial No.** | **Software/Tool** | **Version** | **User for:** |
| 1 | Microsoft SQL Server Management Studio | 18.3.1 | To design database model/diagram and to draw entity diagram to show relationship between tables/views |
| 2 | Microsoft SQL Server 2016 | 13.0.4259.0 | Database to persist and to enforce referential and data integrity |
| 3 | Microsoft Visual Studio 2017 | 15.9.17 | IDE to develop ASP.NET web application |
| 4 | .NET Framework | 4.7.0.3062 | Version 4.7.2, MVC Architecture style/model |
| 5 | Microsoft IIS |  | Runtime for ASP.NET web application |
| 6 | Active Directory |  | Used for authentication |

Chapter 3

DATABASE DESIGN

A database diagram by name “ActivityTracker” has been created using SQL Server Management Studio (SSMS).

This application has following four SQL Server database tables

* Student

To store student’s N# and related attributes.

* ActivityItem

To capture information related to each event that the student participated.

* Semester

A lookup table to capture terms such as Fall 2019, Summer 2019 etc. This table is preloaded with all terms.

* Majors

A lookup table to keep majors/subjects which is also preloaded.

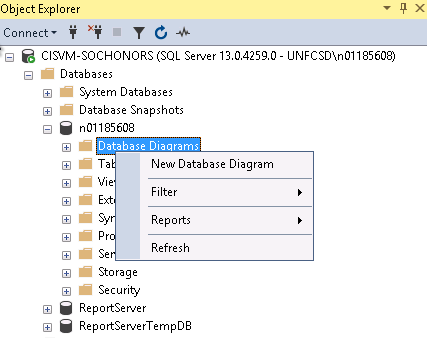
There are three one-to-many relationships defined involving N-Number, Term and major to maintain data integrity across the tables. A view is created as query table, primarily for user interfaces so as not to query tables often in various module for the same information.

Below are the tables along with their attributes. Tables are created using SSMS by right clicking on Tables -> “New Table”.

|  |  |
| --- | --- |
| \ N\_Number is the primary key in “Student” table | Term is the primary key in “Semester” table. |
| Major is the primary key in “Student” table | Primary key,ActivityID, is auto incremented sequence number. |

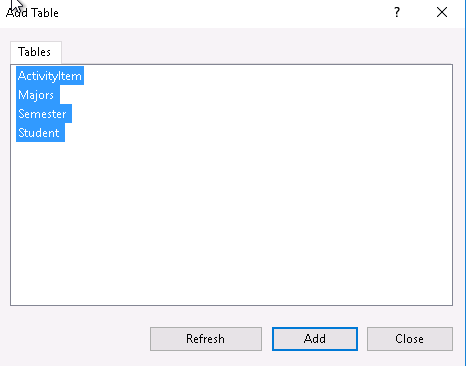
**Figure 1**

A new database diagram is created by selecting above four tables. Right click on “Database Diagrams” -> New Database Diagram



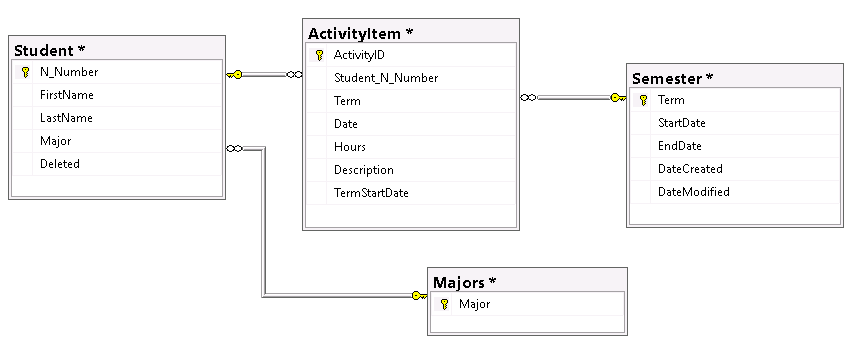
**Figure 2**

From the popup window, select Student, Majors, Semester and ActivityItem



**Figure 3**

Rearrange tables and create relationships and constraints as shown below:



**Figure 3. Entity-Relationship diagram in SSMS**

4

The constraints are:

Student.N\_Number -> ActivityItem.Student\_N\_Number

Student.Major -> Majors.Major

ActivityItem.Term -> Semester Term

ActivityItem is the key table in which student’s activities are to be tracked are stored. It has two one-to-many relationship; 1) N\_Number in ActivityItem to N-Number in Student table and 2) Term in ActivityItem to Term in Semester table.

Assumption : “Current Semester” is determined by the semester with the latest StartDate in Semester table.

**Triggers**: Triggers are created on Semester table upon insert and update to handle DateCreated and DateModified columns.

CREATE TRIGGER insert\_DateCreated  
ON Semester  
FOR INSERT  
AS  
UPDATE Semester  
SET DateCreated = GETDATE()

CREATE TRIGGER update\_DateModified

ON Semester  
FOR UPDATE  
AS  
UPDATE Semester  
SET DateModified = GETDATE()

**Views**: One of the requirements is to build an UI page to show each student’s “current semester” activity hours and total hours. These are derived values based on number of activity rows in

5

ActivityItem and based on latest “Term” from Semester table. Apart from this, UI should exclude soft deleted rows from Student table. A view is created to accomplish this goal. This view has up to date information of all students enrolled in this application.

CREATE VIEW StudentView

AS

SELECT Student.N\_Number, Student.FirstName, Student.LastName, Student.Major, TotalActivityCount, TotalHours, CurrentSemesterActivityCount, CurrentSemesterHours, sortDate

FROM Student

OUTER APPLY

(

SELECT SUM(ActivityItem.Hours) as TotalHours, count(\*) as TotalActivityCount

FROM ActivityItem

WHERE ActivityItem.Student\_N\_Number = Student.N\_Number

) AS T1

OUTER APPLY

(

SELECT SUM(ActivityItem.Hours) as CurrentSemesterHours, count(\*) as CurrentSemesterActivityCount

FROM ActivityItem

WHERE ActivityItem.Student\_N\_Number = Student.N\_Number and

ActivityItem.Term = (SELECT TOP 1 term FROM Semester ORDER BY StartDate DESC)

) AS T2

OUTER APPLY

(

SELECT TOP 1 ActivityItem.TermStartDate as sortDate

FROM ActivityItem

WHERE ActivityItem.Student\_N\_Number = Student.N\_Number

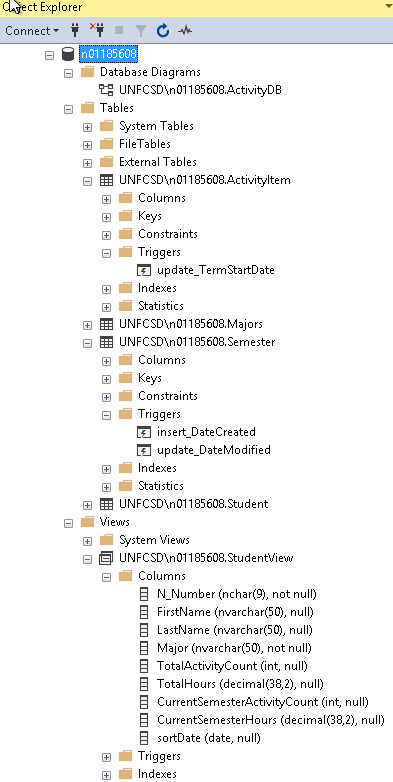
ORDER BY ActivityItem.TermStartDate DESC

) AS T3

WHERE Student.Deleted IS NULL

Upon saving the database diagram, the triggers and views, the SSMM Object explorer will look like:

6



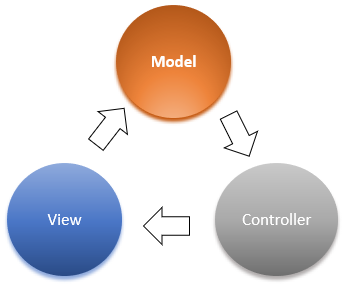
**Figure 5**

7

Chapter 4

ARCHITECTURE PATTERN

Model-View-Controller (MVC) is one of the widely used architecture pattern which separates an application into three major components; Models, Views and Controllers. This pattern ensures the separation of concerns; the model represents the data, view represents the UI of data and controller manages model data and views. Controller interprets the user interactions on the View (UI) and orchestrates the workflow of views based on user interactions in each of the view. The separation of concerns is that model data is independent of views and controllers; views are independent of model data and controllers. The controller is inserted in between the models and views to remove the model-view dependencies. In MVC pattern, the controller is the entry point and it collects the data from model objects, applies validations, business rules and presents the data to the user as Views.



**Figure 6. Model-View-Controller**

ASP.NET has three built-in development models: 1) Web Pages 2) Web Forms and 3) MVC (Model-View-Controller). ASP.NET MVC framework is the chosen pattern for this project as it

8

is integrated well with the existing ASP.NET features, such as Authorization and Authentication, Master Pages, Data Binding, User Controls, Memberships, ASP.NET Routing, etc., It has in-built Razor and helper classes. All these features make application development with ASP.NET MVC comparatively easier. The below outlined features are used in this project.

**Routing**:

APS.NET MVC is built on top of ASP.NET routing which provides convention-based routing that the application accepts and map to a specific action method within a given controller.

public async Task<IActionResult> Login(LoginViewModel model, string returnUrl = null) { ... }

**Model binding**:

APS.NET MVC model binding converts form values, route data, query strings parameters into objects that the controller can handle. As a result, controller is presented with the data as parameters and action methods.

routes.MapRoute(name: "Default", template: "{controller=Home}/{action=Index}/{id?}");

**Model validation**:

This feature makes data validation done at the client side before controller action is called on server side. Many of the edits and data validations can be added to model class itself as strongly typed helper methods decorations. In the controller, ModelState.IsValid can be checked before applying business logic.

9

**Filters**:

Filters enables action methods to run at certain conditions such as exception handling or authorization. Filters can be applied globally to controllers or to actions as attributes. Active directory authorization is enforced in this project by applying filters in each controller.

[Authorize]

public class AccountController : Controller

ASP.NET MVC uses Razor view engine to render views along with strongly typed model decorations in entity classes. This means controller can pass strongly typed model to views to do type checking and IntelliSense support such as IEnumerable<ActivityItems>.

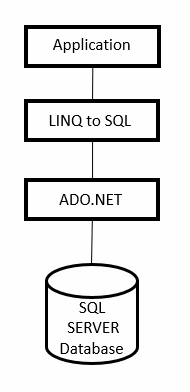
10

Chapter 5

ADO.NET ENTITY FRAMEWORK

Entity Framework is a data access technology that supports data-oriented application development. Entity framework exposes interfaces in creating entity data models (EDM), relationships between them (ERM) and business logic. It also interfaces with data engines to store and to retrieve data, and thus providing database access and data operations. Microsoft’s entity framework, ADO.NET, makes entity-relationship modelling executable. ADO.NET uses xml files, auto generated codes in the form of .NET objects and ADO.NET APIs. The ADO.NET Entity Framework, which is an Object-Relational Mapping (ORM) framework, allows to write C# programs against classes that are auto generated from the conceptual model and takes care of translations between the code and backend databases. It has a number of abstraction layers to provide data access with ease of wizards and drag-and-drop controls. This project uses ADO.NET framework to import data models created in Microsoft SQL Server Management studio (SSMS) and to auto generate data entity classes and to enforce relationships between them. ADO.NET framework allows to write programs against the model and against the database tables. This framework sits between the code and the database and provides objects that closely matches the business instead of rows and columns. The advantage of using any Entity framework, in this case ADO.NET, is that it shields the database implementation from application development. At the time of execution, the respective database provider can be selected as long as database has implemented open source object-relational mapping specifications. LINQ (Language Integrated Query) is a set of extensions to .NET Framework which along with ADO.NET extends C#’s ability to do data operations with various data sources much easier.

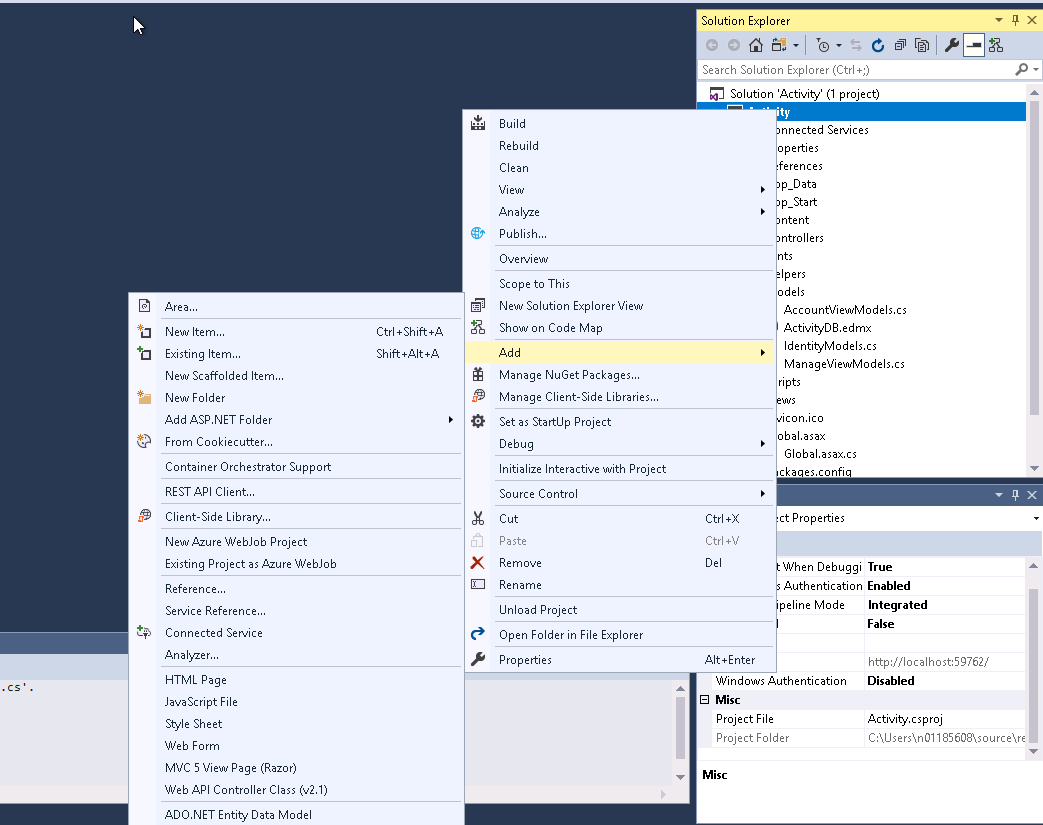
11

  
 **Figure 7**

This project uses MS SQL Server provider, System.Data.EntityClient, which is in web.config as part connectionString.

The database model that is created using SSMS is then added/imported into Visual Studio project as a data item using ADO.NET Entity Data Model framework which would create four c# classes representing each table; student.cs, semester.cs, major.cs and activityitem.cs.

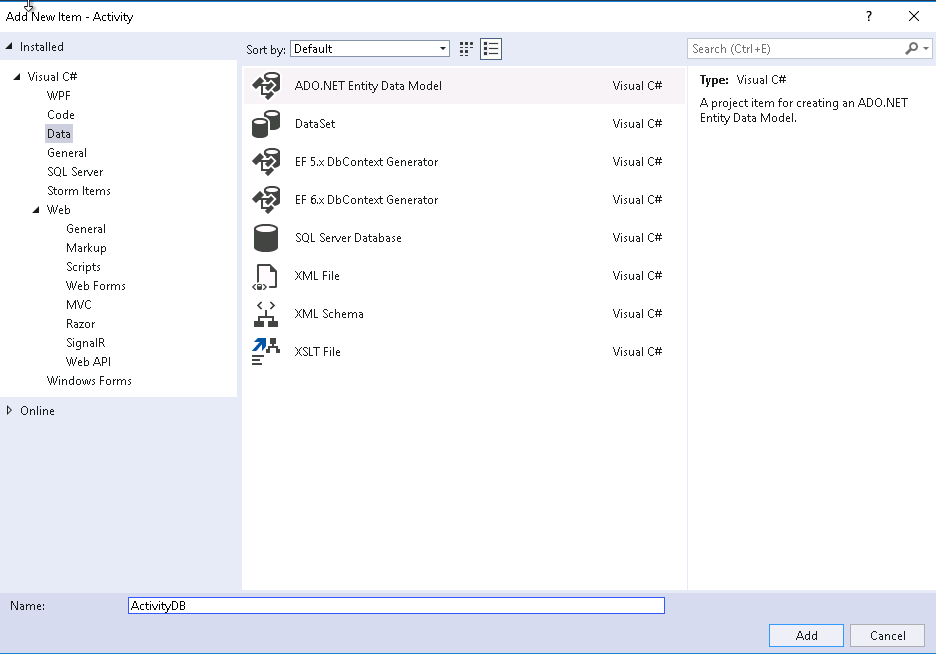
Right Click on the Solution (“Activity” solution) in Visual Studio -> Add -> New Item



**Figure 8**

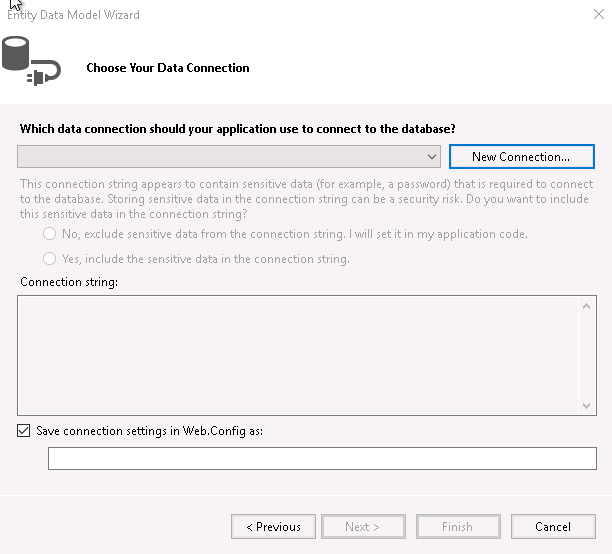
Select Data -> ADO.NET Entity Data Model with Type Visual C#

12



**Figure 9**

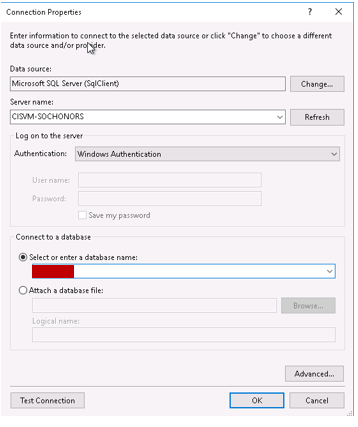
Click on New Connection (to connect to SQL Server database instance)



**Figure 10**

13

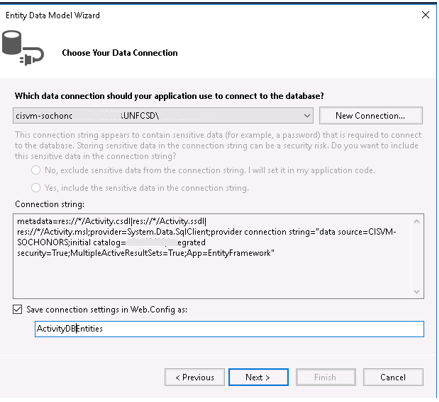
Enter the SQL Server Host Name and select the Database as shown below:



**Figure 11**

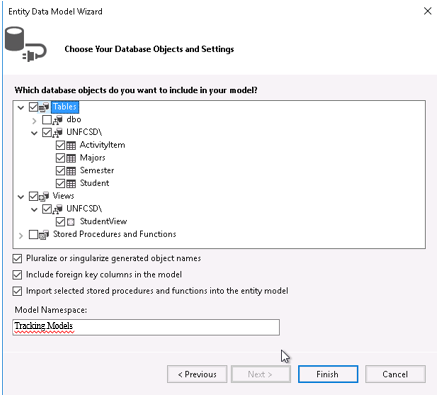
Upon click on “OK”, it will bring up a with SQL Server Connection string to be added to solution’s web.config file.

14



**Figure 12**

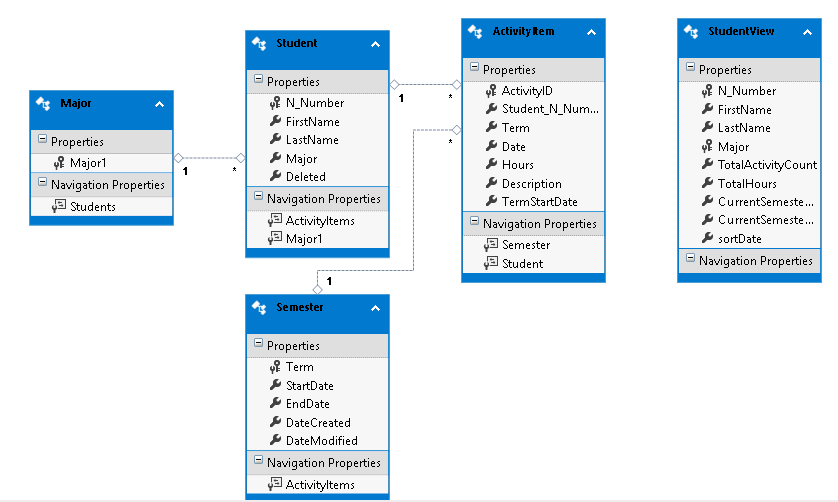
In the next window, select tables and views



**Figure 13**

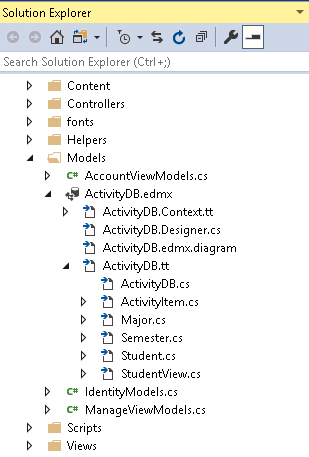
15

Click on Finish. This will create a model based on database design on SSMM and created ER diagram along with C# classes for each of tables and views.



**Figure 14**

The generated models and entity classes can be viewed and/or edited in Visual Studio Solution Explorer.



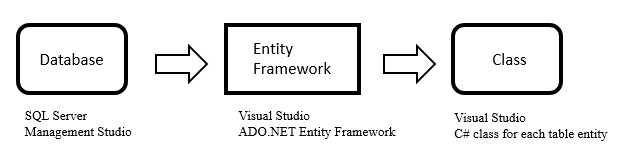
**Figure 15**

16

Chapter 6

DESIGN

ADO.NET MVC with Visual Studio built-in functions supports three different design approaches and they are 1) code first 2) Model first and 3) Database first. Database first approach has been chosen for this project.



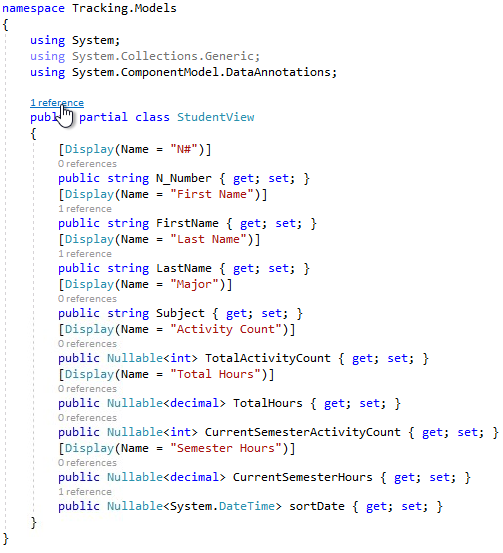
**Figure 16**

Database modelling and design using Microsoft SQL Server Management studio (SSMS) is discussed in Chapter 3. Once Database is created and validated in SSMS, it is then added/imported as ADO.NET Entity Framework item in Visual Studio (Chapter 5). This procedure, upon completion, creates ER diagram, auto generates C# classes for each one of the tables along with backend SQL Server connection string in web.config.

The auto-generated data entity classes, Student, Semester, Major and ActivityItems are decorated with strongly typed helper method to do basic validation at UI level before the data passed to the controller. Once such example of strongly typed entity class is shown below:

17





18



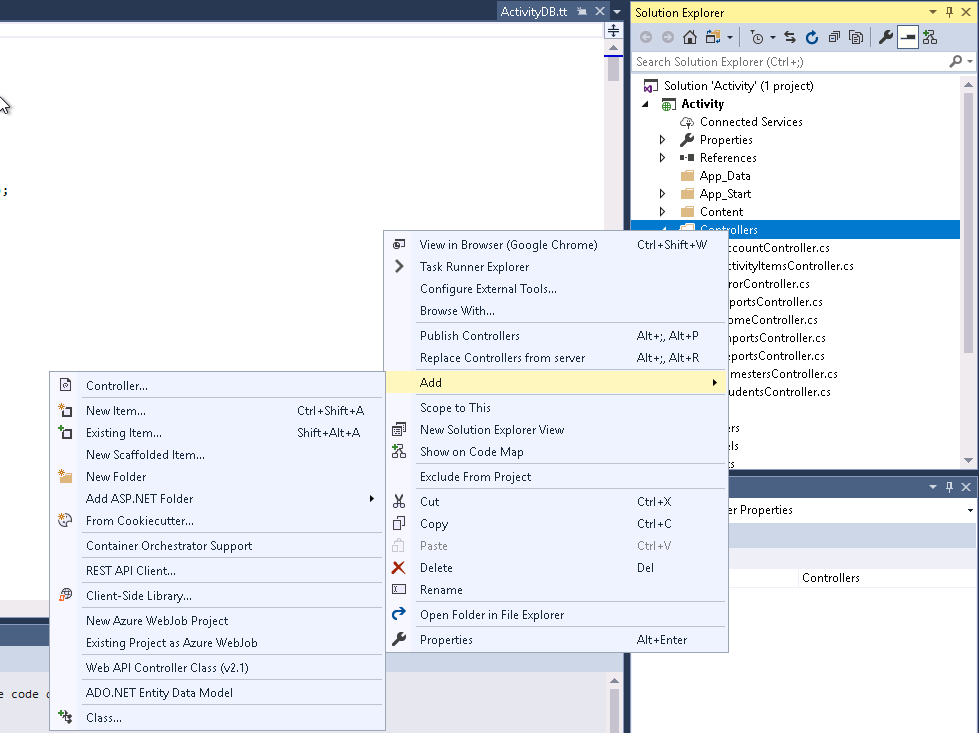
**Figure 17**

Visual Studio has built-in pre-installed code generators called Scaffolding, a code generation framework. Scaffolding quickly allows to add a template code to the solution that interacts with data models/classes created by ADO.NET framework. In MVC design pattern, a controller can be added using scaffolding by choosing a model class with default views.

Below are the steps to create Controller and Views for Semester table/Entity:

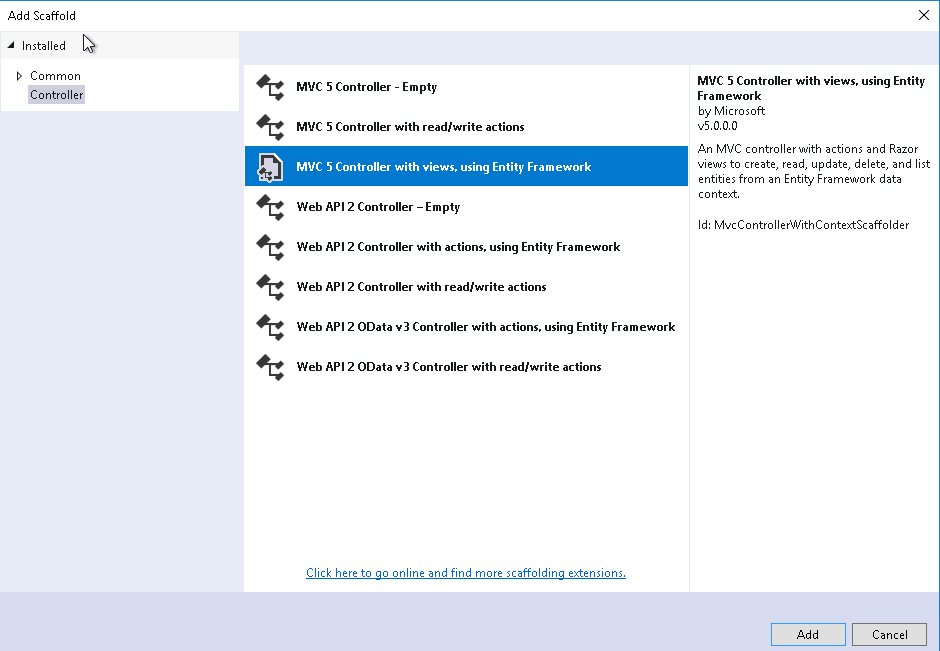
In the Solution Explorer, right click on Controllers -> Add -> Controller

19



**Figure 18**

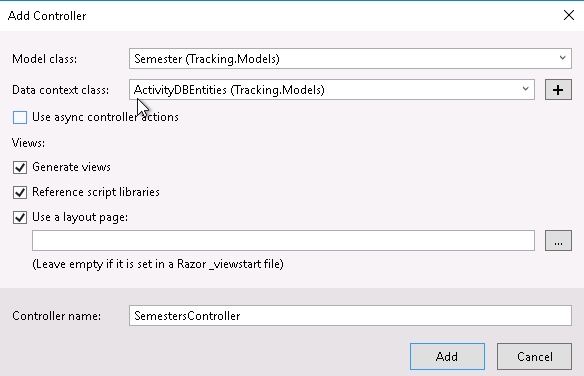
Select “MVC 5 Controller with views, using Entity Framework”



**Figure 19**

Select the Model class for which Controller to be created(Ex: Semester) and check Generate Views as shown below:

20



**Figure 20**

Upon completion of Scaffolding, Controller and its Views will be created.



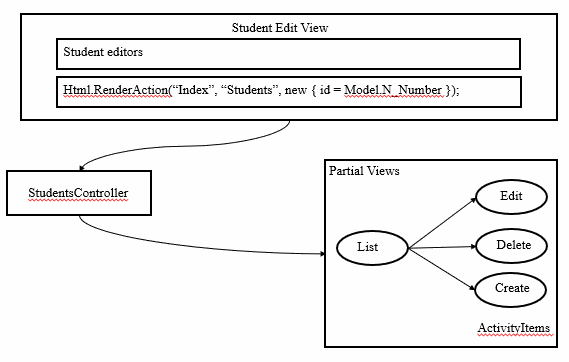
**Figure 21**

The same procedure is then followed to the rest of the classes; Student, Major & ActivityItems.

The generated controller code, by default, will have Index, Create, Detail, Delete methods along with optional view for each one of the controller methods. The default naming convention would be the same as entity model class name. The generated codes and views are then modified to implement data validations, business rules/logic and the default behavior. Since generated controller code and cshtml views will work with one data model (by default), partial views are created to handle multiple data entity classes in one view as is the case in this project. The

21

general approach taken in the project is to have a view, for an example “Students” Index view with partial views from other data entity classes, for an example ActivityItems \_List view. This makes user interaction less complex by keeping all relevant data in one single view. This eliminates the need for users to switch views back and forth.



**Figure 22**

To handle the partialViews in the controller:

Instead of:

return View(“Index”, await ActivityItems.ToListAsync());

change has been made as below:

return PartialView(“\_Index”, await ActivityItems.ToListAsync());

The built-in HTML Helper classes are used in every view. Some of the helper classes are:

@Html.Label

@using  
@Html.DropDownList  
@Html.Hidden  
@Html.TextBox

22

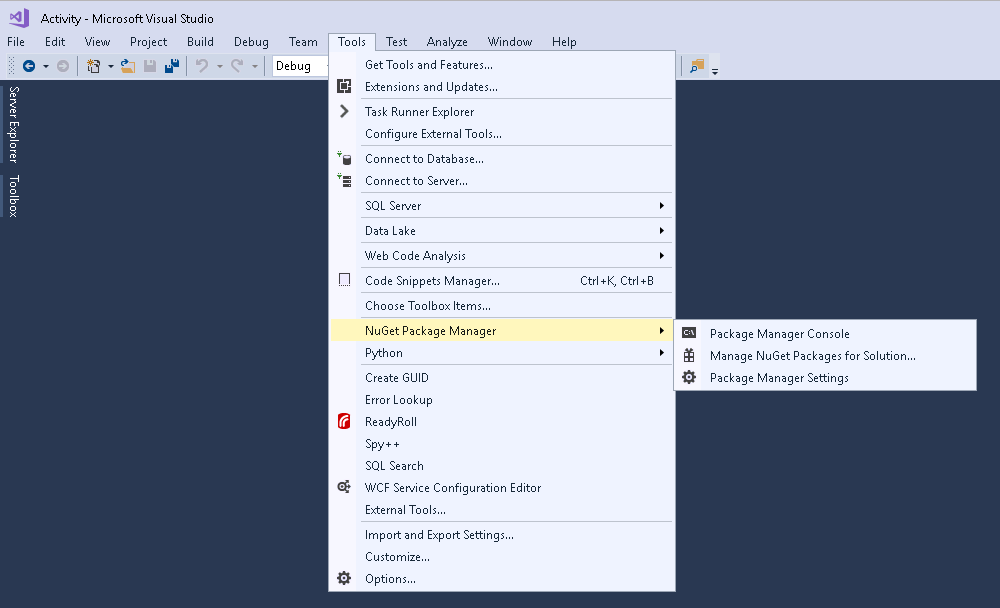
Chapter 7

DATA EXPORT AND IMPORT - EXCEL

At times, it is required to export data into Excel file to validate and/or for record keeping. Batch or bulk data load may be required instead of keying in data one at a time; this requires a way to import data from Excel file into database tables. Though there are many ways to export to and import from Excel file into tables, closedXML is being used in this application. closedXML is a .NET library which provides easy interfaces to read, write and manipulate Excel files. It is built upon openXML API.

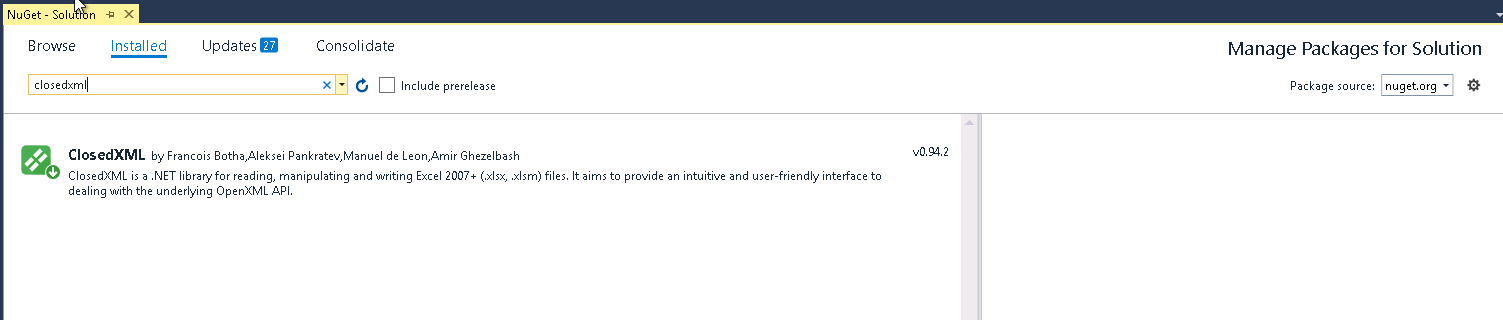
closedXML is installed in Visual Studio using NuGet package Manager.

Select Tools -> NuGet Package Manager -> Manage NuGet Packages for Solution



**Figure 23**

Search for “closedXML” and install.



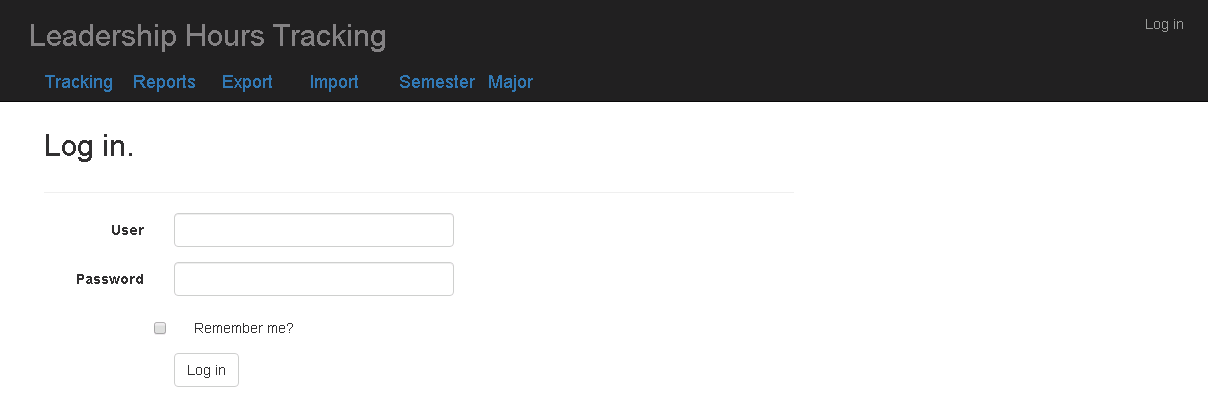
**Figure 24**

23Chapter 8

SECURITY

It is a general practice to secure a web application that is not meant for general public consumption. The data exposed in the application could be sensitive and they should be protected and managed by those who are authorized. Before granting access to the application UI, identity/credential of the user is collected in the form of user & password and cross checked against a central identity manager. Active Directory (AD) is one such software which organizes and provides identity and access information. Active Directory maintains roles, groups and users in a hierarchically structured directory fashion. Leadership Hours tracking application is secured by AD authentication and authorization.

Every controller is decorated with [Authorize] attribute which forces the Login page, if not already validated, to obtain user’s credential which is then validated against AD in LoginAccountController.



**Figure 25**

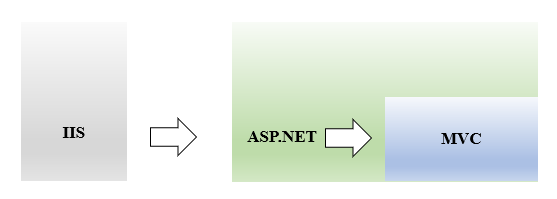
Note: Authentication & Authorization hasn’t been tested at the time of writing this document, pending permission and access to AD.

24

Chapter 9

INTERNET INFORMATION SERVICES (IIS)

Internet Information Services (IIS) is Microsoft’s Web server to host web applications. ASP.NET is a framework to process requests originating from any sources but primarily uses IIS (Internet Information Services). The flow of request is shown below in Figure 26.

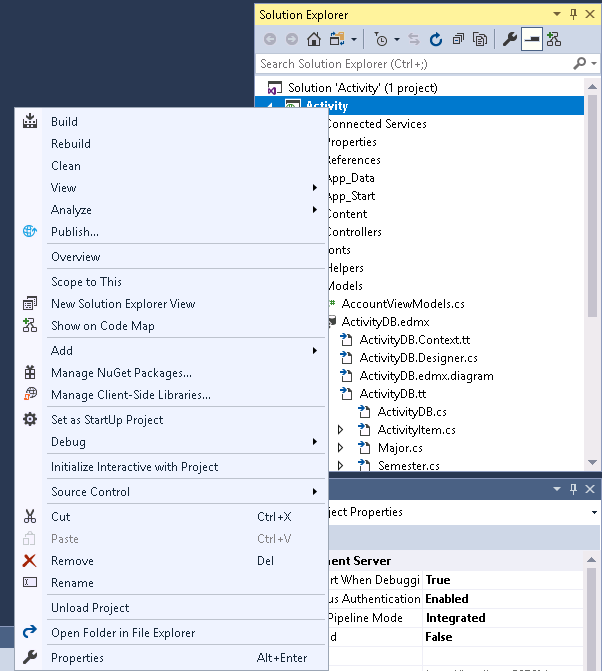


**Figure 26**

Any type of handlers can be registered with ASP.NET of which HttpHandlers of file type cshtml is one of them. By default , ASP.NET does not handle cshtml requests forwarded by IIS. By having MVC registered as HttpHanlder, ASP.NET will forward all cshtml requests to MVC. The “Leadership Hours Tracking” solution project has been published to IIS default website folder on CISVM-SOCHONORS server at c:\inetpub\wwwroot.

To publish the solution to IIS, right click on the solution and select “Publish”.

25



**Figure 27**

From the dropdown, select “FolderProfile” and click on Configure. Select “File System” as Publish method and “c:\inetpub\wwwroot” as Target location. Save the configuration and click on “publish”. <http://localhost> or <http://CISCM-SOCHONORS> will bring up the default web application.

26

Chapter 10

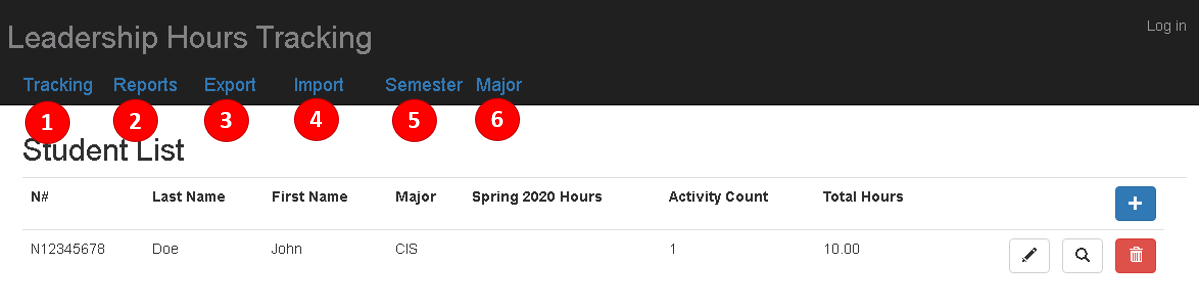
WALKTHRU

“Leadership Hours Tracking” Application can be launched by

Option 1: Express IIS in visual studio

Option 2 : ) <http://CISVM-SOCHONORS> in the browser as it is published to the IIS default site at wwwroot.

Landing page:  
The landing page lists all students and their hours for the current semester, activity count, and their total hours.



1. Tracking

Tracking is the landing page.

1. Reports

To list all the activities for a given search criteria using filters.

1. Export

To export activities into the Excel file based on a given search criteria using filters.

1. Import

To import activities from an Excel file into the ActivityItem table.

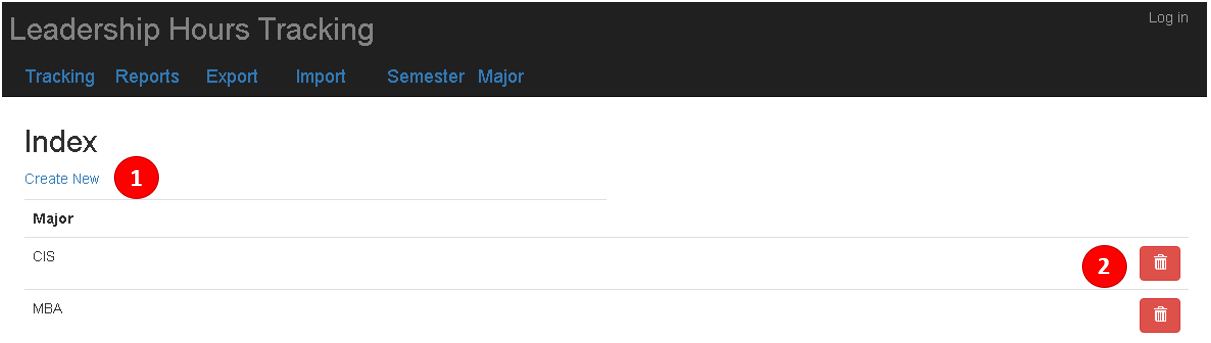
1. Semester

To manage terms

1. Major

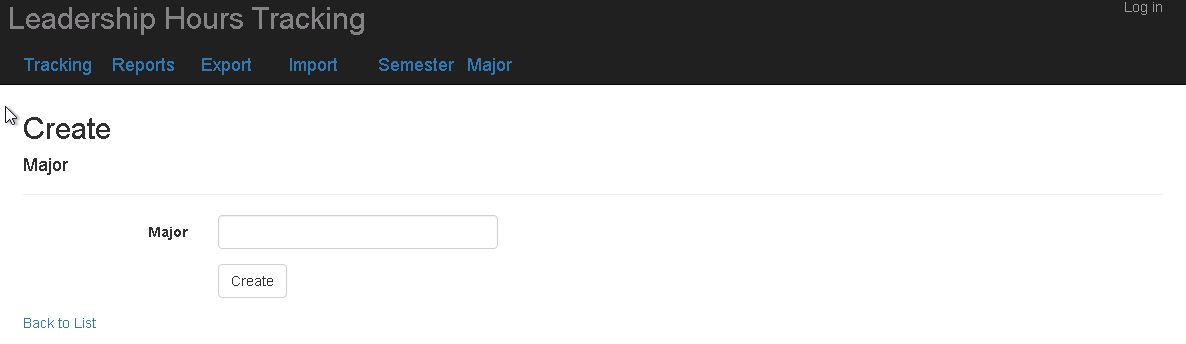
To manage majors.

**Majors:**



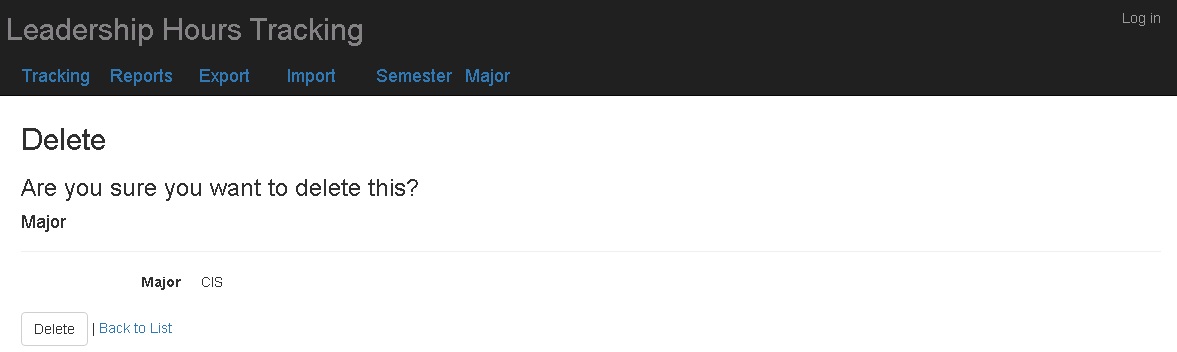
Major view lists all the majors.

1. Create New: To add new major



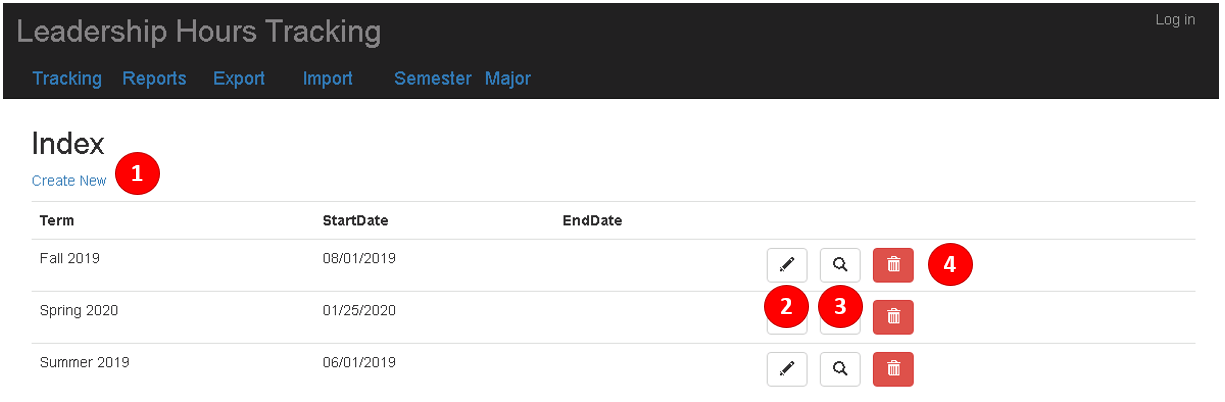
Enter major and click on create. “Back to List” takes you back to the list view.

1. Delete: To delete a major

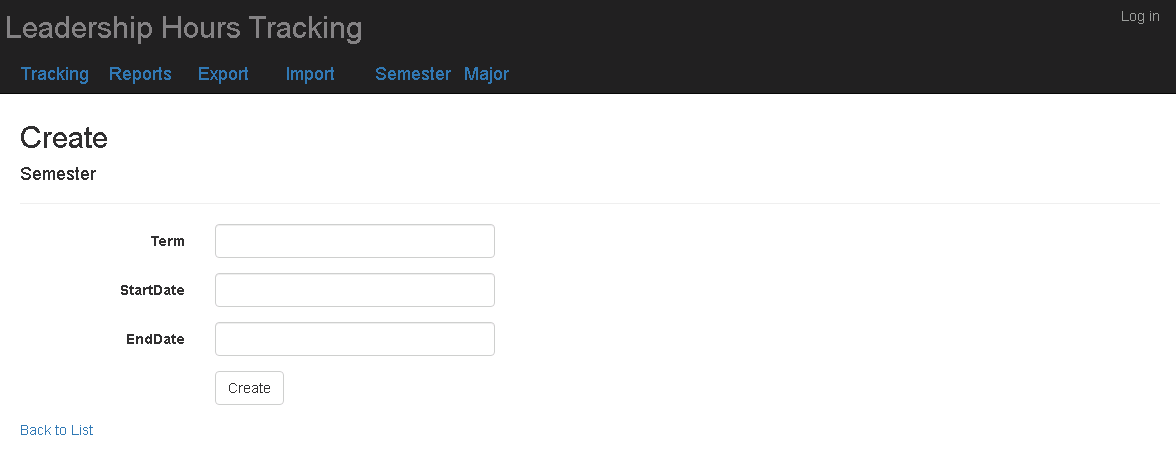


Click on “Delete” to delete the major. It cannot be deleted if there are students registered with this major.

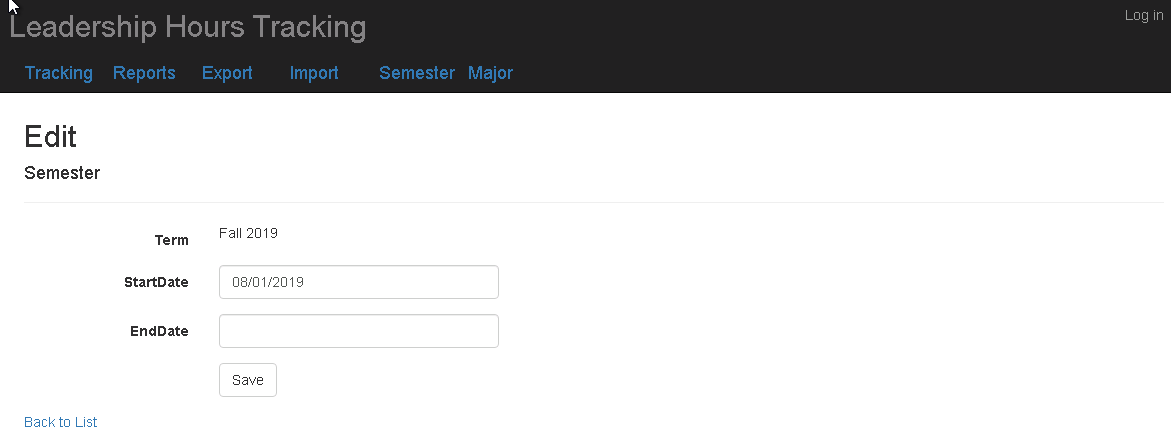
**Semester:**



1. Create New: To add new semester/term

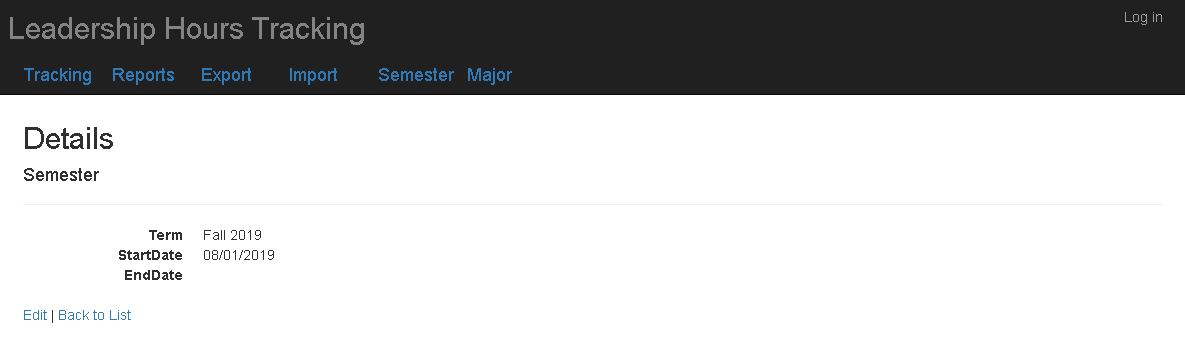


Enter Term and start/end date and click on create. Start Date is a required field, an accurate Start Date must be entered in order for the “Current Semester Hours” to be displayed properly. “Back to List” takes you back to the list view.

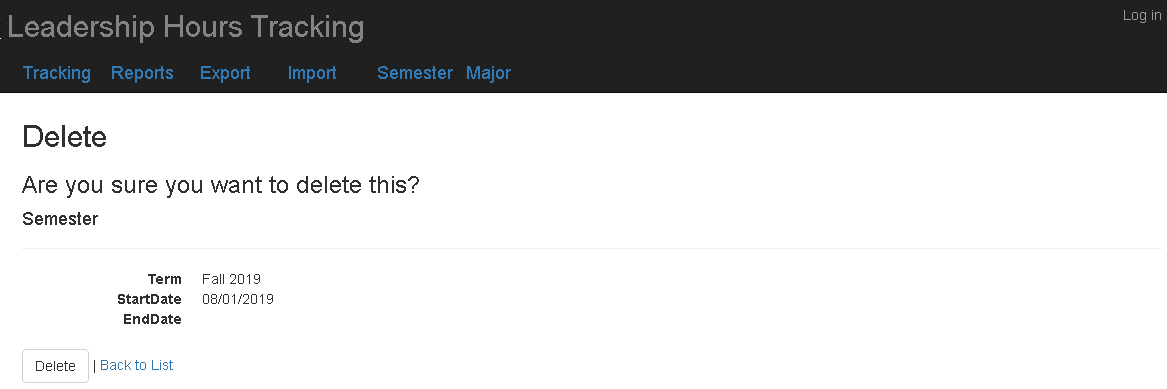
1. Edit: To edit/update a Term

Click on “Save” to save changes.

1. Edit: To query a Term

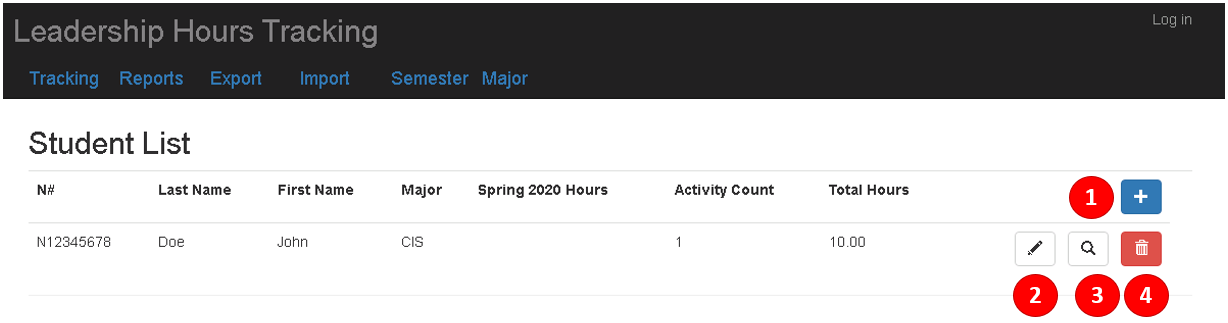


1. Delete: To delete a Term

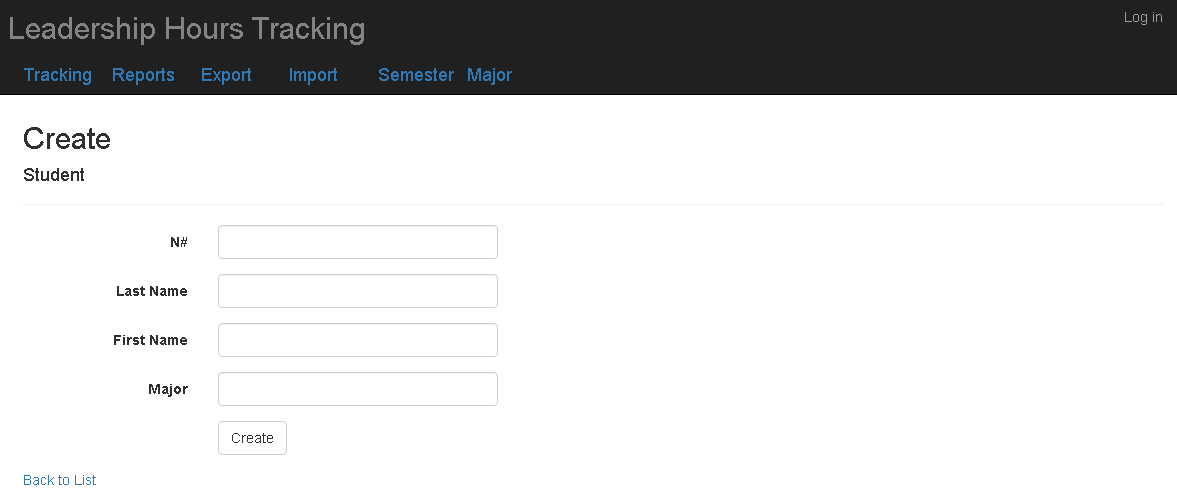


Click on “Delete” to delete the Term. It cannot be deleted if there are activities for this term.

**Tracking:**

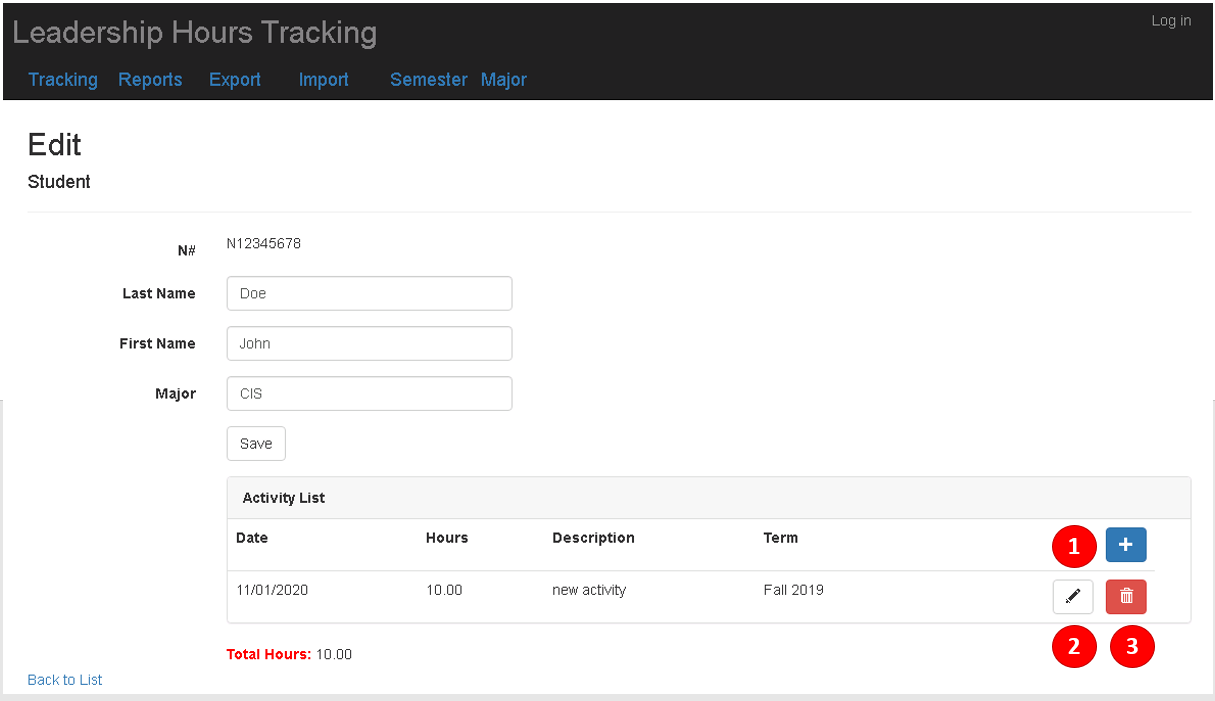


1. Create New (+): To register a new student.



Enter student information and click on “Create”. N# and Major are required fields. “Back to List” takes you back to the list view.

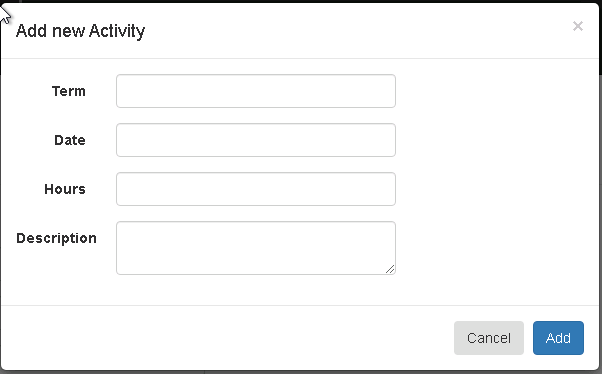
1. Edit: To edit/update student information.



Change/Update student information and click on “Save”.

1. To add new activity to the student, click on 

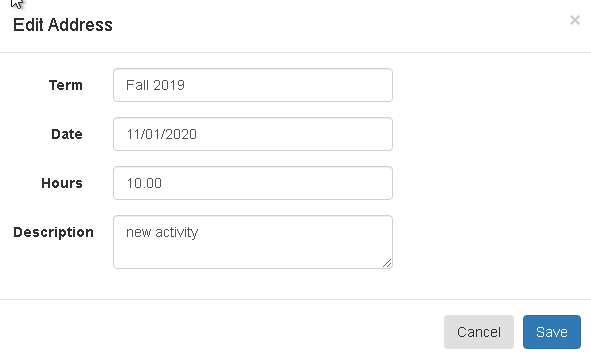
This will bring up a pop-up window



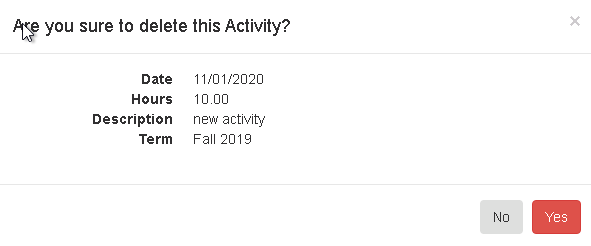
Enter Activity information and click on “Add”. Term and Hours are required fields.

1. To edit an activity, click on 

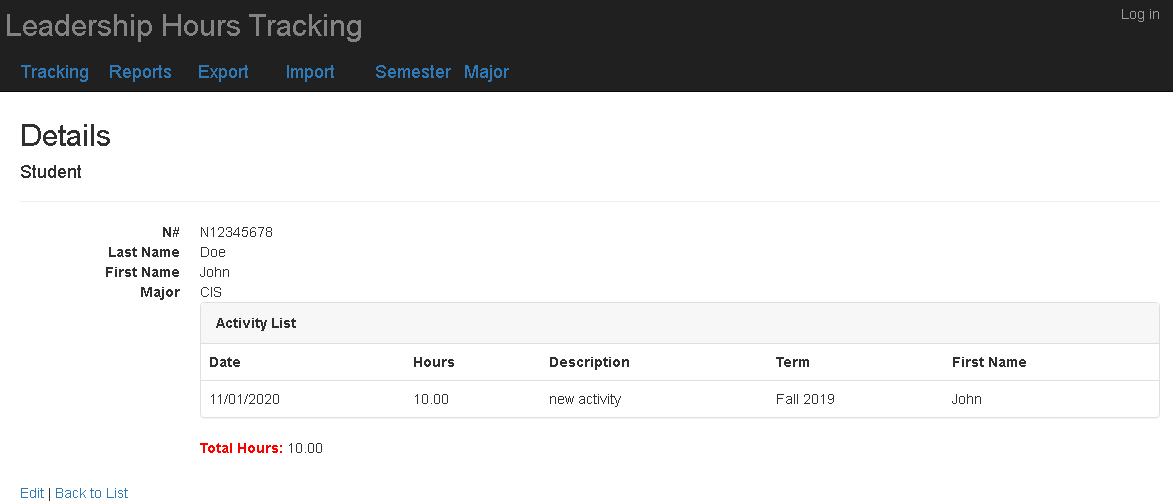
Change/Update activity information on the pop-up window, click on “Save”

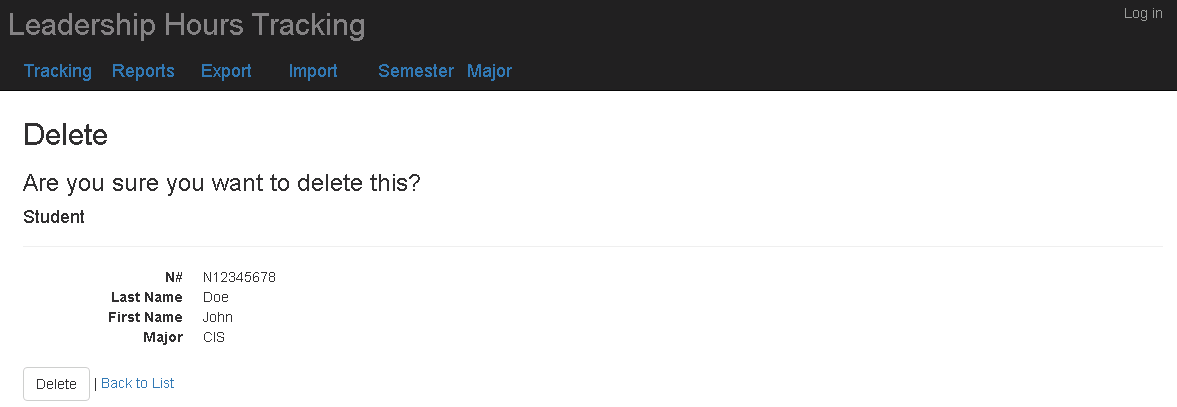


1. To delete an activity, click on 



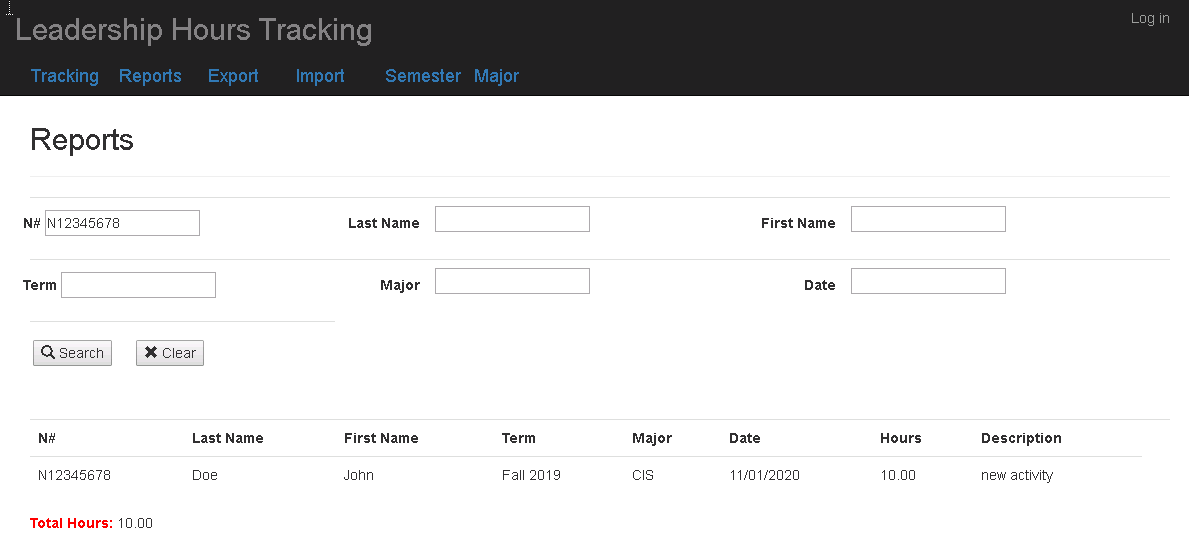
Click on “Yes” to confirm.

1. To query student information.
2. To delete/unregister a student.



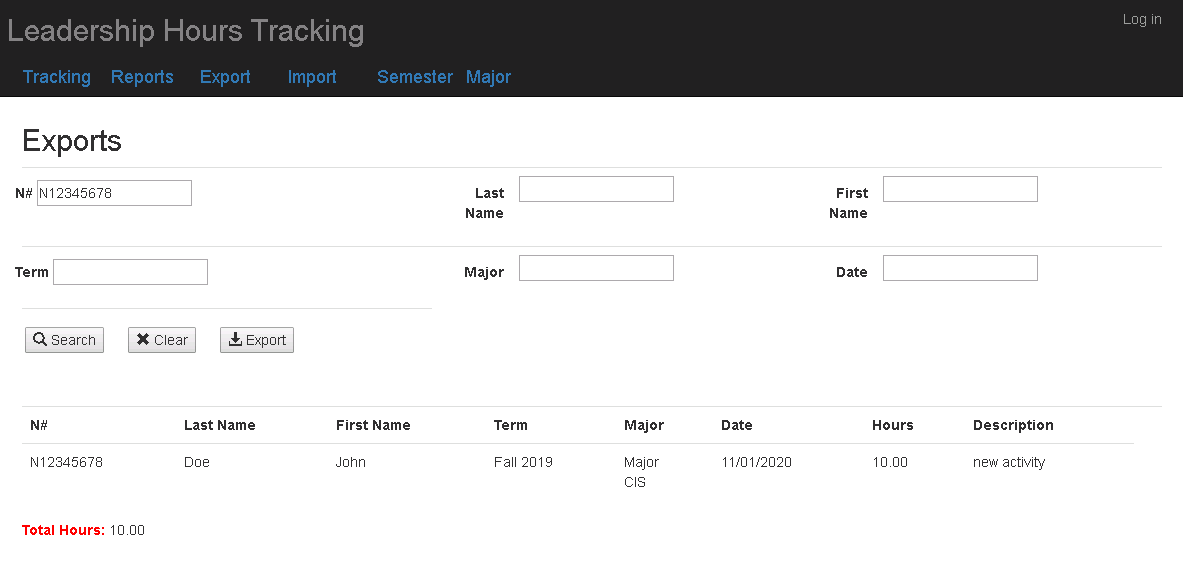
Click on “Delete” to do the “soft” delete on student. The “Deleted” column will be updated to “1” and the student will be unlisted from any changes. In order to add the Student once again, the database must be manually updated by changing the “Deleted” column for that student back to “NULL”.

**Reports:**

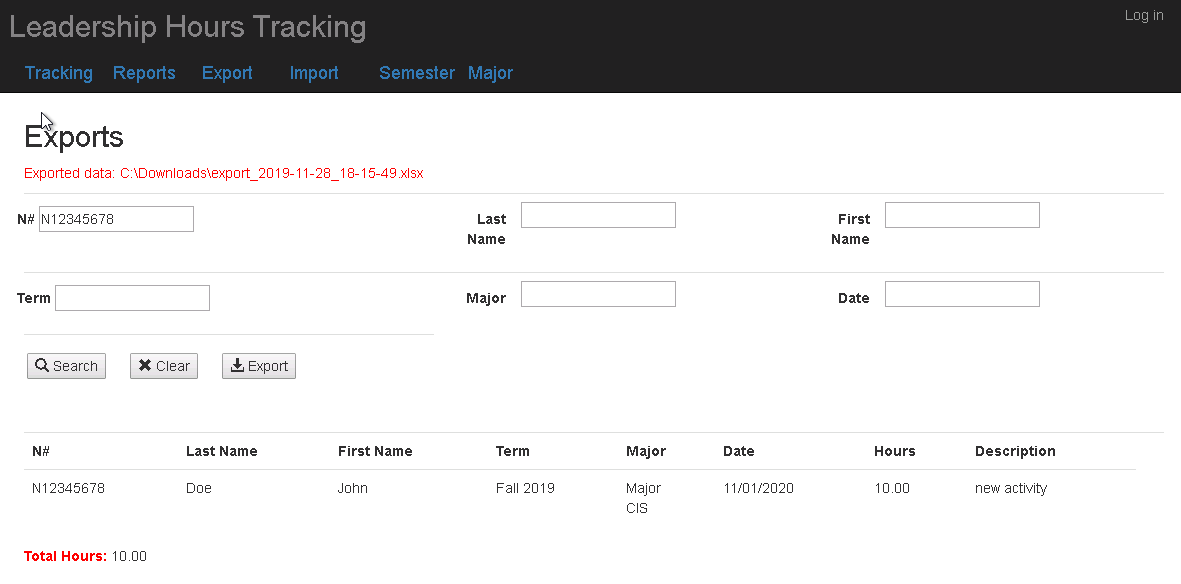


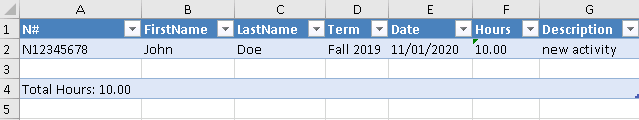
By default, it displays activities of all students. Enter filter conditions to narrow the list and click on “Search”. To clear the search criteria/filters, click on “Clear”.

**Export:**

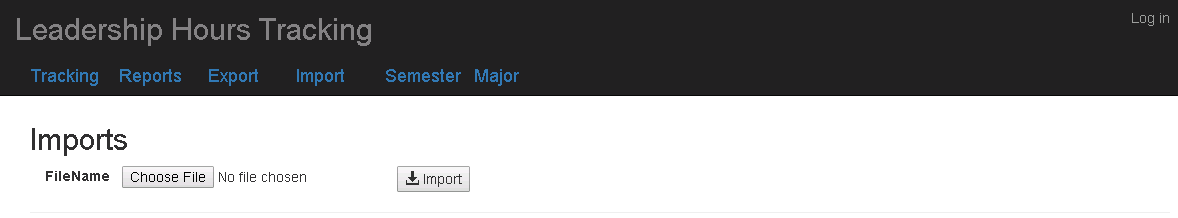


By default, it displays activities of all students. Enter filter conditions to narrow the list and click on “Search”. To clear the search criteria/filters, click on “Clear”. To export the data into Excel file, click on “Export”. The exported file name will be displayed at the top of the page as well as selected activities.



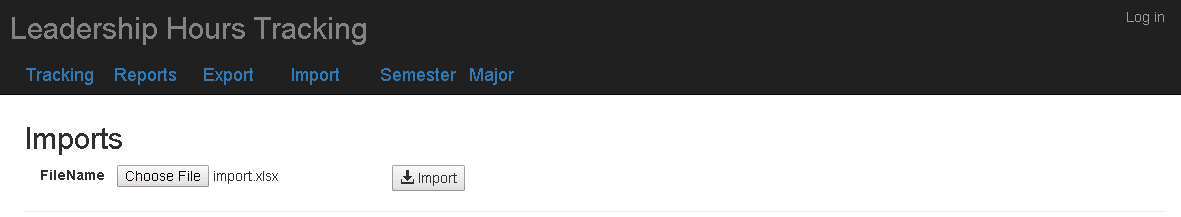


**Import:**

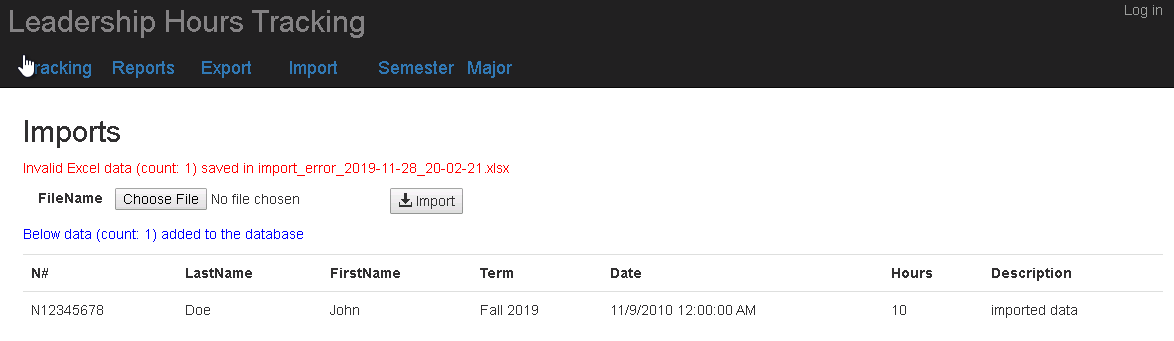


Click on “Choose File” to select the Excel file that contains activity data.





Click on “Import”



It will display the number of imported rows as well as the imported data. Invalid data will be written into another excel file for users to correct and reimport.

APPENDIX A: DELIVERABLE

**Week 1: October 18 thru 25th**

Wire frames and basic database schema/design. Delivered via electronic document (.docx or pdf)

See Eligible data entities and their properties of concern for list of entities and candidates.

Connect to server and ensure connectivity. (Instructions coming)

**Week 2: October 25th thru November 1st.**

First user screen delivered and working. This screen will allow user to enter in data.

See page (x) for details of requirement.

**Week 3: November 1st thru November 8th**

Demonstrate the ability to provide a report from the application.

See page (x) for requirement for report.

**Week 4: November 8th thru November 15th**

Ad integration.

Ensure authentication to application is in place.

**Week 5: November 15th thru November 22nd**

Add ability to import and or export data from excel forms.

Establish required formats etc.

**Week 6: November 22nd thru end of term**

Provide documentation of various required backend code. Documentation should include the functionality of the various objects/methods/functions and scripts including the data they rely on and a description of what they do. Versioning requirements should be included.

37

APPENDIX A: DATABASE SCHEMA

CREATE TABLE [UNFCSD\<database>].[ActivityItem](

[ActivityID] [int] IDENTITY(1,1) NOT NULL,

[Student\_N\_Number] [nchar](9) NOT NULL,

[Term] [nvarchar](20) NOT NULL,

[Date] [date] NULL,

[Hours] [decimal](4, 2) NULL,

[Description] [nvarchar](250) NULL,

[TermStartDate] [date] NULL,

CONSTRAINT [PK\_ActivityItem] PRIMARY KEY CLUSTERED

(

[ActivityID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

CREATE TABLE [UNFCSD\<database>].[Semester](

[Term] [nvarchar](20) NOT NULL,

[StartDate] [date] NOT NULL,

[EndDate] [date] NULL,

[DateCreated] [datetime] NULL,

[DateModified] [datetime] NULL,

CONSTRAINT [PK\_Semester] PRIMARY KEY CLUSTERED

(

[Term] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

CREATE TABLE [UNFCSD\<database>].[Student](

[N\_Number] [nchar](9) NOT NULL,

[FirstName] [nvarchar](50) NULL,

[LastName] [nvarchar](50) NULL,

[Subject] [nvarchar](50) NOT NULL,

[Deleted] [nchar](1) NULL,

CONSTRAINT [PK\_Student] PRIMARY KEY CLUSTERED

(

[N\_Number] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

CREATE VIEW [UNFCSD\<database>].[StudentView]

AS

SELECT Student.N\_Number, Student.FirstName, Student.LastName, Student.Subject, TotalActivityCount, TotalHours, CurrentSemesterActivityCount, CurrentSemesterHours, sortDate

FROM Student

OUTER APPLY

(

SELECT SUM(ActivityItem.Hours) as TotalHours, count(\*) as TotalActivityCount

FROM ActivityItem

WHERE ActivityItem.Student\_N\_Number = Student.N\_Number

) AS T1

38

OUTER APPLY

(

SELECT SUM(ActivityItem.Hours) as CurrentSemesterHours, count(\*) as CurrentSemesterActivityCount

FROM ActivityItem

WHERE ActivityItem.Student\_N\_Number = Student.N\_Number and

ActivityItem.Term = (SELECT TOP 1 term FROM Semester ORDER BY DateCreated DESC)

) AS T2

OUTER APPLY

(

SELECT TOP 1 ActivityItem.TermStartDate as sortDate

FROM ActivityItem

WHERE ActivityItem.Student\_N\_Number = Student.N\_Number

ORDER BY ActivityItem.TermStartDate DESC

) AS T3

WHERE Student.Deleted IS NULL

CREATE TABLE [UNFCSD\<database>].[Majors](

[Subject] [nvarchar](50) NOT NULL,

CONSTRAINT [PK\_Majors] PRIMARY KEY CLUSTERED

(

[Subject] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

ALTER TABLE [UNFCSD\<database>].[ActivityItem] WITH CHECK ADD CONSTRAINT [FK\_ActivityItem\_Semester] FOREIGN KEY([Term])

REFERENCES [UNFCSD\<database>].[Semester] ([Term])

ALTER TABLE [UNFCSD\<database>].[ActivityItem] CHECK CONSTRAINT [FK\_ActivityItem\_Semester]

ALTER TABLE [UNFCSD\<database>].[ActivityItem] WITH CHECK ADD CONSTRAINT [FK\_ActivityItem\_Student] FOREIGN KEY([Student\_N\_Number])

REFERENCES [UNFCSD\<database>].[Student] ([N\_Number])

ALTER TABLE [UNFCSD\<database>].[ActivityItem] CHECK CONSTRAINT [FK\_ActivityItem\_Student]

ALTER TABLE [UNFCSD\<database>].[Student] WITH CHECK ADD CONSTRAINT [FK\_Student\_Majors] FOREIGN KEY([Subject])

REFERENCES [UNFCSD\<database>].[Majors] ([Subject])

ALTER TABLE [UNFCSD\<database>].[Student] CHECK CONSTRAINT [FK\_Student\_Majors]

39

REFERENCES

ASP.NET MVC

<https://dotnet.microsoft.com/apps/aspnet/mvc>

<https://docs.microsoft.com/en-us/aspnet/mvc/overview/getting-started/introduction/getting-started>

<https://sensibledev.com/mvc-bootstrap-form-example/>

<https://www.tutorialspoint.com/asp.net_mvc/asp.net_mvc_overview.htm>

C#

<https://docs.microsoft.com/en-us/dotnet/csharp/>

<https://www.tutorialspoint.com/csharp/index.htm>

<https://www.c-sharpcorner.com/>

Authentication:  
<http://www.codedigest.com/posts/4/forms-authentication-using-active-directory-users-in-aspnet-mvc>

<https://stackoverflow.com/questions/10279140/configure-asp-net-mvc-for-authentication-against-ad>

Excel – ClosedXML:

<https://github.com/ClosedXML/ClosedXML>

Glyphicons:

<https://getbootstrap.com/docs/3.3/components/>

ADO.NET:

<https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/ado-net-overview>

IIS:

<https://www.iis.net/>

<https://www.youtube.com/watch?v=nrlayNIFd8s>

SQL Server Management Studio:

<https://www.codeproject.com/Articles/1185300/Creating-A-SQL-Server-Database-From-Visual-Studio>

40