

Software Engineering Course nline Marking Tool

December 2015

Diploma of Software Engineering and Design - Final Project Submission for Heilene Daem

Table of Contents

[Objective 1](#_Toc440181230)

[Overview 1](#_Toc440181231)

[Functionallity and features 1](#_Toc440181232)

[Development Process 2](#_Toc440181233)

[Technologies used for development: 2](#_Toc440181234)

[Phase One – to be completed by 11 December 2015: 2](#_Toc440181235)

[Phase Two – to be completed by 28 February 2016: 2](#_Toc440181236)

[Project Schedule 3](#_Toc440181237)

[Client briefing and overview 3](#_Toc440181238)

[User Experience Design Flow and Database Design 4](#_Toc440181239)

[Workflow wireframe 4](#_Toc440181240)

[Database design 5](#_Toc440181241)

[Implementation 6](#_Toc440181242)

[Build relational database structure 6](#_Toc440181243)

[Build Student, Module, Overview comments, overtitles MVC 6](#_Toc440181244)

[Basic MVC scaffolding of the Students, Modules and Enrolments 7](#_Toc440181245)

[Completed view layouts 8](#_Toc440181246)

[Student List Page 8](#_Toc440181247)

[Student Details Page 8](#_Toc440181248)

[Marking Tool View 9](#_Toc440181249)

[Edit View page 9](#_Toc440181250)

[Results Page 10](#_Toc440181251)

[Build Marking Tool View Page – created a ViewModel with DatabaseCall class and Data Transfer Object class 11](#_Toc440181252)

[View Model Code: 11](#_Toc440181253)

[Build Result View Page - Edit option for lecturer – view only for student 13](#_Toc440181254)

[Build Edit View Page 14](#_Toc440181255)

[Troubleshoot 16](#_Toc440181256)

[Rubric Table set up 16](#_Toc440181257)

[MarkSet Values 16](#_Toc440181258)

[Deleting Related Data 18](#_Toc440181259)

# Objective

## Overview

Development of an online marking tool for the Software Engineering and Design course for Vision College that is an electronic copy of the current marking document which is currently a Microsoft office word document.

Each module of the course, contains a course overview, marking overview and an assessment PDF.

The marking tool will reflect the current document and includes the marking standards for each module attributed to the course. The standards include a Marking Overview and Rubric Table.

The tool will allow the user/lecturer to mark student assessments online after which students can access their course module marks.

Technologies to build project includes ASP.NET MVC/Razor, Entity Framework, SQL Management Server Studio and will be hosted on Azure for testing.

Initially the tool will be utilized and tested by the Christchurch campus then deployed for use at the Hamilton and Auckland campuses.

## functionallity and features

Lecturers will be able to use this tool to add, edit and delete students and create and add modules which is part of the Software Engineering Course.

Each module includes a unique Marking. The module also includes a Rubric Marking Table that is the same for each module.

Marking of assessments will be done online and saved for each student

Students will be able to access their results online as well as download the assessments that are available for each module.

Students will be able to submit a link to complete assessments which are cloud base hosted or on GITHUB.

Students will have limited access and will only be able to view results and submit assessments.

Lecturers will have full site access.

# Development Process

## technologies used for development:

* Visual Studio 2015
* ASP.NET MVC 5
* Entity Framework
* Microsoft SQL Studio Management Server
* Azure

## Phase One – to be completed by 11 December 2015:

* Database Design and Implementation
* Webpage which allow user to add, delete and edit Software Engineering Students
* Webpage which allow user to add, delete and edit Modules assigned to students
* Basic MVC scaffold which allows lecturer to update Module Marking Overview titles and comments
* An interactive marking page that enables lecturer to add marks based on current document layout
* Result layout page, with editing feature
* Students are assigned modules upon creation of a new student

## Phase Two – to be completed by 28 February 2016:

* Update Module Detail view to include Marking Overview details
* Update Module Edit page to allow user to update marking overview and overview titles
* Login Authorization to Students and Lecturers with appropriate authority application
* Students can upload link to assignment files whether it’s to their GitHub repository, Google Drive, Dropbox or any cloud storage system
* Apply theme and artwork to project
* Implement ability to add Assignment PDF per Module document for download
* Upload project to Azure

# Project Schedule

Client briefing and initial design

|  |  |
| --- | --- |
| Description | completion date |
| Client Briefing | 26/10/2015 |
| Submit Project Proposal | 30/10/2015 |
| Database Design and UX Design Submission | 04/11/2015 |
| Signed-off Design | 06/11/2015 |
| Implementation and submission of phase one | 09/11/2015 – 10/12/2015 |
| Implementation and submission of phase two | 20/01/2016 – 28/02/2016 |

## client briefing and overview

I had a meeting with the Software Engineering Lecturer Gary Dix to discuss the functionality of the marking tool. Gary supplied me with all the documentation which included all the course module layouts, assessments, a copy of the current marking document.

I have enlisted mentoring from Gary Dix and the majority of the project will be completed onsite. I will also be using services as Stack Overflow and Google.

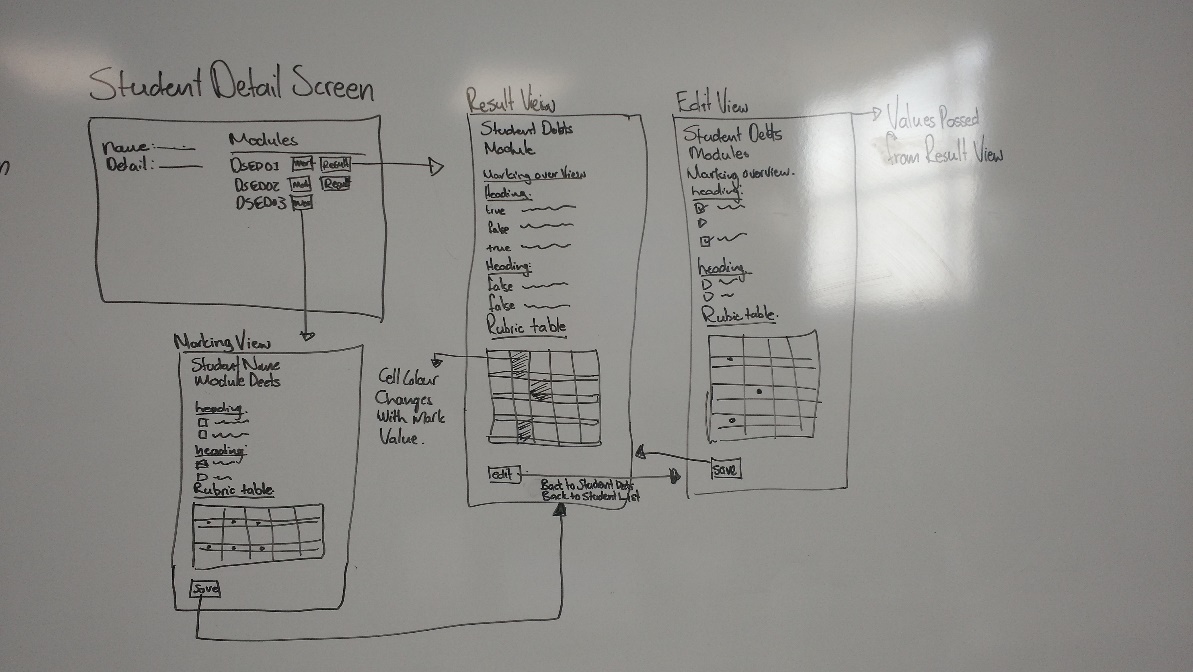
I am hoping to gain an in-depth knowledge of ASP.NET MVC/Razor, protocol to upload to server.

# User Experience Design Flow and Database Design

### Workflow wireframe

The landing page will give the user the option to the student list and module list. From the student list, once a student has been added and has been attributed modules, the marking view can be accessed from the student detail page.

The student detail page as demonstrated below displays the Modules together with a Mark button. It will display a result button if a result ID is available.

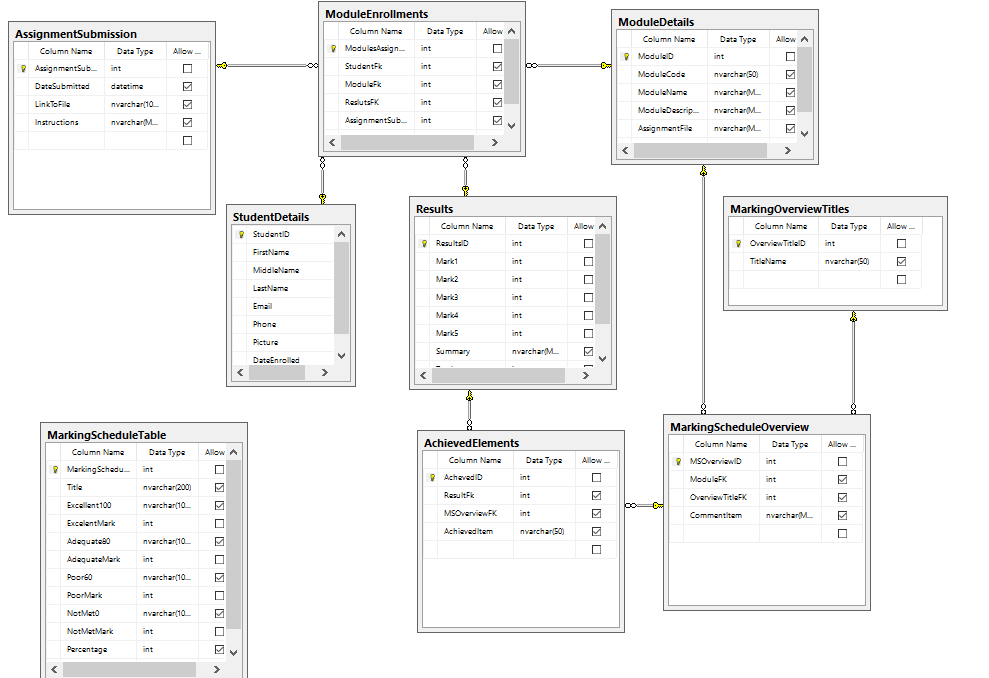


## 

## Database design

I spent some amount of time working on my relational database for the project. As this is the integral part of my project I wanted to get it as accurate as possible.

My original data base had the Marking Overview table with 10 comments per row. As part of database normalisation I reconfigure the database tables for AchievedElements and MarketingScheduleOverview which cut down significantly on repeated code as well as simplifying the database insert for achieved elements per result. I am pleased that I did make the decision to do this as it will save a lot of coding time. This took me about 1.30 hours to update and introduce back into my project.

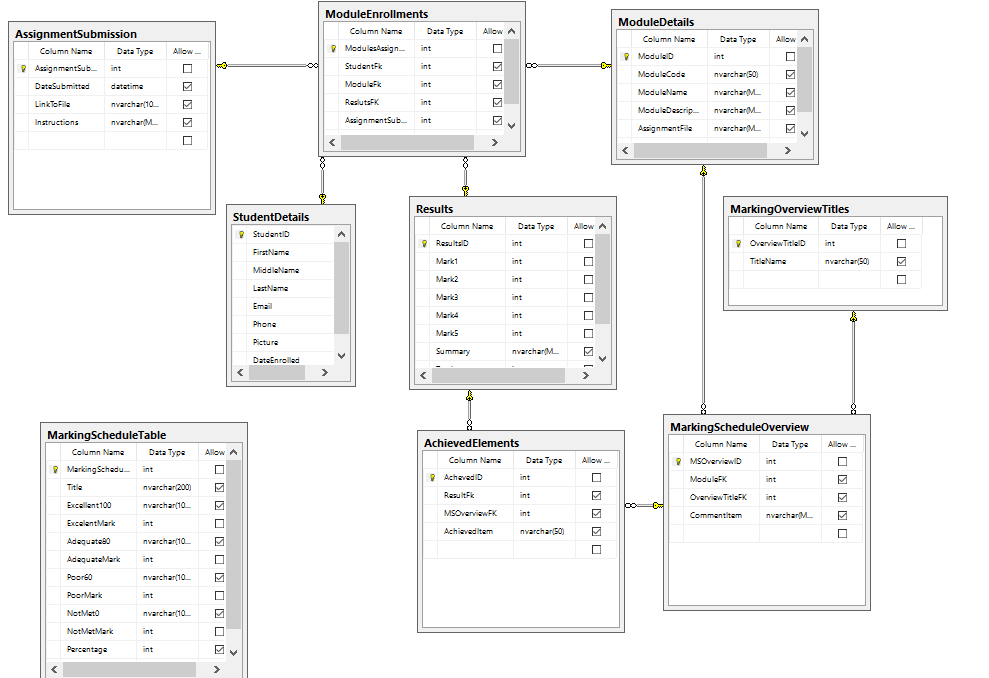


# Implementation

## Build relational database structure

I started by drawing my design on paper. My initial design had the connection between students and module with an assignment table. This design element caused some issues with the relation between the table and the updating of related tables. I fixed this by creating a Module Enrolment table which held foreign keys for Students, Modules, and Results.

The AssignmentSubmission Table will be accessed in phase two, when implementing the functionality of students having the ability to submit assignment links.



## Build Student, Module, Overview comments, overtitles MVC

I used ASP.NET to scaffold the Module View Controllers for these tables. This automatically generated the functionality to update, delete and create students, modules, overview comments and titles. I then modified the student detail view to display the modules assigned to each student.

Phase two, I will modify the Module display to show the overview comments and titles per module. Then create the ability to update this display in a single view.

## C:\Users\DWX\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2015-12-10_1237.pngBasic MVC scaffolding of the Students, Modules and Enrolments

I’ve used ASP.NET MVC 5 to automatically create the scaffold pages for the StudentDetail, ModuleDetail. ModuleEnrollments, MarkingScheduleOverview and MarkingOverviewTitles Tables.

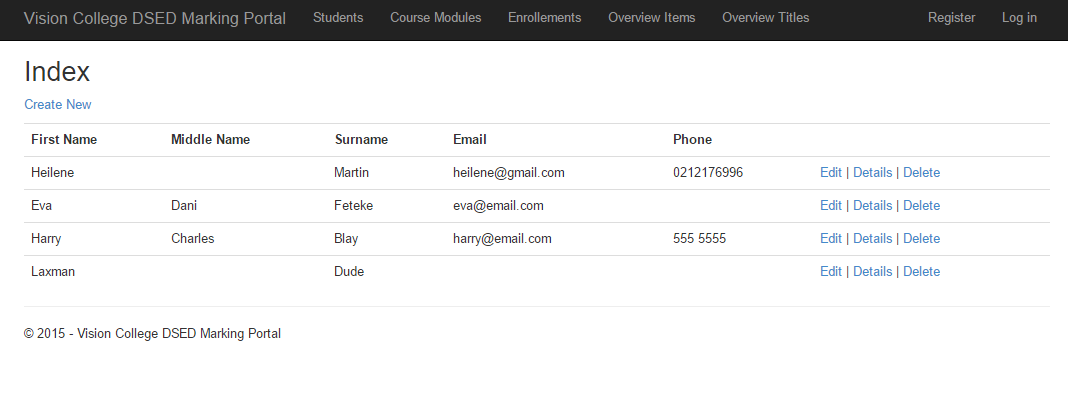
The key table here is the StudentDetails Table. This is where the user can delete edit and create students. I have also added the Marking View, Result View and Edit Marking View under the StudentDetails Controller.

I have included the Overview and Title tables as these will need editing capability as Modules change and new Modules are created.

## Completed view layouts

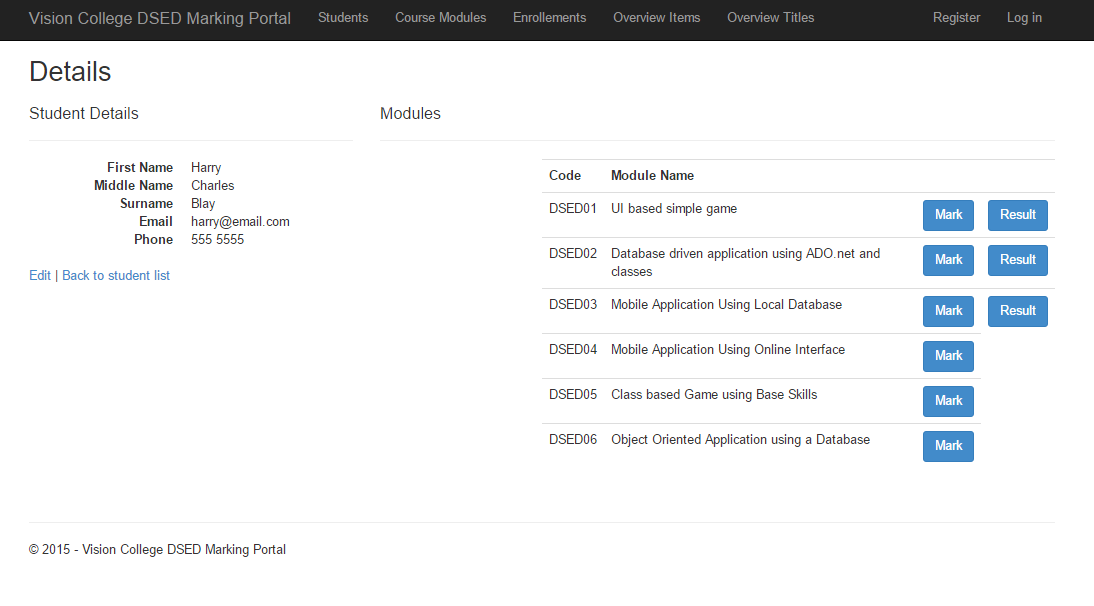
### Student List Page

Click through to student details, edit, create or delete



### Student Details Page

Click through to either marking view or result view. The layout has been edited to include the module code and name assigned to each student. The mark button clicks through to the module marking page. As module marking is saved a result ID is generated which adds the Result button to the student detail page.



### Marking Tool View

Save click display Result page

### Edit View page

Same as the Marking view, with   
student result values pulled through   
from database. Save click to Result  
page.

### Results Page

Click through to edit view,   
back to student details or  
back to student list

Build Marking Tool View Page – created a ViewModel with DatabaseCall class and Data Transfer Object class

* Pull Module Code and Name from Module ID
* Pull Student FullName and ID
* Marking Overview Table
* Generate Rubric Table using Database
* Create update calls to database – Update Results, Enrolments and Achieved Elements

### View Model Code:

//GET:StudentDetail/MarkingToolView/5

//how to pass values to controller http://stackoverflow.com/questions/8646944/persist-values-between-two-action-results-in-one-controller

[HttpGet]

public ActionResult MarkingToolView( int? id, int? studentID)

{

MarkingVM MyMarkingVM = new MarkingVM();

ResultsDTO MyResults = new ResultsDTO();

// StudentMarkingDTO MyStudentData = new StudentMarkingDTO();

//return student deets

MyResults = myDatabaseCalls.getStudentDetails(studentID, id);

MyMarkingVM.StudentID = studentID;

MyMarkingVM.ModuleID = id;

//return marking overview

MyMarkingVM.MyMarkingOverview = myDatabaseCalls.GetMarkingOverview(id);

MyMarkingVM.MarkAchievedList = myDatabaseCalls.GetMarkingOverview(id);

MyResults.ModuleID = id;

//get unique titles of overview for display

MyMarkingVM.MyOverviewTitles = myDatabaseCalls.GetUniqueTitle(id);

//return marking rubric table value per row index

MyMarkingVM.MyTableRow1 = myDatabaseCalls.GetRubricTable(1);

MyMarkingVM.MyTableRow2 = myDatabaseCalls.GetRubricTable(2);

MyMarkingVM.MyTableRow3 = myDatabaseCalls.GetRubricTable(3);

MyMarkingVM.MyTableRow4 = myDatabaseCalls.GetRubricTable(4);

MyMarkingVM.MyTableRow5 = myDatabaseCalls.GetRubricTable(5);

//return Mark values for radio button

MyMarkingVM.myMarksSet1 = myDatabaseCalls.GetMarkSetValues1();

MyMarkingVM.myMarksSet2 = myDatabaseCalls.GetMarkSetValues2();

MyMarkingVM.myMarksSet3 = myDatabaseCalls.GetMarkSetValues2();

MyMarkingVM.myMarksSet4 = myDatabaseCalls.GetMarkSetValues3();

MyMarkingVM.myMarksSet5 = myDatabaseCalls.GetMarkSetValues3();

//Viewbags

ViewBag.ModuleDeets = string.Format("{0} - {1}", MyResults.MCode, MyResults.MName);

ViewBag.StudentDeets = string.Format("{0}", MyResults.SFullName);

return View(MyMarkingVM);

}

//POST:StudentDetail/MarkingToolView/Create

[HttpPost]

public ActionResult MarkingToolView(MarkingVM MarkingModel)

{

if (ModelState.IsValid)

{

//Get Marking total value

MarkingModel.MarkTotal = MarkingModel.GetTotalMark(MarkingModel.MyResults);

//update results Database

myDatabaseCalls.CreateNewResultsRecord(MarkingModel);

return RedirectToAction("ResultView", new { id = MarkingModel.ModuleID, studentID = MarkingModel.StudentID});

}

return View(MarkingModel);

}

## 

## Build Result View Page - Edit option for lecturer – view only for student

* Marking Overview Table
* Generate Rubric Table using Database
* Pull all student ID Module ID

//GET: StudentDetails/ResultView/5

public ActionResult ResultView(int? id, int? studentID)

{

ResultVM ResultViewModel = new ResultVM();

ResultsDTO MyResultDetails = new ResultsDTO();

//return student deets

ResultViewModel.MyResults = myDatabaseCalls.GetStudentResults(studentID, id);

ResultViewModel.StudentID = studentID;

ResultViewModel.ModuleID = id;

//Get mark total

//ResultView.MarkTotal = ResultView.GetTotalMark(ResultView.MyResults);

//return marking overview and Marks Achieved Values

ResultViewModel.MyMarkingOverview = myDatabaseCalls.GetMarkingOverview(id);

//get unique titles of overview for display one title per comment set

ResultViewModel.MyOverviewTitles = myDatabaseCalls.GetUniqueTitle(id);

//return marking rubric table value per row index

ResultViewModel.MyTableRow1 = myDatabaseCalls.GetRubricTable(1);

ResultViewModel.MyTableRow2 = myDatabaseCalls.GetRubricTable(2);

ResultViewModel.MyTableRow3 = myDatabaseCalls.GetRubricTable(3);

ResultViewModel.MyTableRow4 = myDatabaseCalls.GetRubricTable(4);

ResultViewModel.MyTableRow5 = myDatabaseCalls.GetRubricTable(5);

//get table cell background colours

myDatabaseCalls.GetBackGroundValues(ResultViewModel.MyTableRow1, ResultViewModel.MyResults.RMark1);

myDatabaseCalls.GetBackGroundValues(ResultViewModel.MyTableRow2, ResultViewModel.MyResults.RMark2);

myDatabaseCalls.GetBackGroundValues(ResultViewModel.MyTableRow3, ResultViewModel.MyResults.RMark3);

myDatabaseCalls.GetBackGroundValues(ResultViewModel.MyTableRow4, ResultViewModel.MyResults.RMark4);

myDatabaseCalls.GetBackGroundValues(ResultViewModel.MyTableRow5, ResultViewModel.MyResults.RMark5);

//Viewbags

ViewBag.ModuleDeets = string.Format("{0} - {1}", ResultViewModel.MyResults.MCode, ResultViewModel.MyResults.MName);

ViewBag.StudentDeets = string.Format("{0}", ResultViewModel.MyResults.SFullName);

return View(ResultViewModel);

}

## Build Edit View Page

* Pull Result DTO details into form
* Create Data call to update all relevant tables

//GET:StudentDetail/MarkingToolView/5

//how to pass values to controller http://stackoverflow.com/questions/8646944/persist-values-between-two-action-results-in-one-controller

[HttpGet]

public ActionResult EditMarkingView(int? id, int? studentID)

{

EditVM MyEditVM = new EditVM();

MyEditVM.MyResults = myDatabaseCalls.GetStudentResults(studentID, id);

MyEditVM.ResultID = MyEditVM.MyResults.ResultID;

MyEditVM.ModuleID = MyEditVM.MyResults.ModuleID;

//return marking overview

MyEditVM.MyMarkingOverview = myDatabaseCalls.GetMarkingOverview(id);

MyEditVM.MarkAchievedList = MyEditVM.MyResults.myAchievedList;

MyEditVM.MyResults.ModuleID = id;

//get checked value for AchievedElements

MyEditVM.UpdateCheckedValues(MyEditVM.MarkAchievedList);

//get unique titles of overview for display

MyEditVM.MyOverviewTitles = myDatabaseCalls.GetUniqueTitle(id);

//return marking rubric table value per row index

MyEditVM.MyTableRow1 = myDatabaseCalls.GetRubricTable(1);

MyEditVM.MyTableRow2 = myDatabaseCalls.GetRubricTable(2);

MyEditVM.MyTableRow3 = myDatabaseCalls.GetRubricTable(3);

MyEditVM.MyTableRow4 = myDatabaseCalls.GetRubricTable(4);

MyEditVM.MyTableRow5 = myDatabaseCalls.GetRubricTable(5);

//return Mark values for radio button

MyEditVM.myMarksSet1 = myDatabaseCalls.GetMarkSetValues1();

MyEditVM.myMarksSet2 = myDatabaseCalls.GetMarkSetValues2();

MyEditVM.myMarksSet3 = myDatabaseCalls.GetMarkSetValues2();

MyEditVM.myMarksSet4 = myDatabaseCalls.GetMarkSetValues3();

MyEditVM.myMarksSet5 = myDatabaseCalls.GetMarkSetValues3();

//Viewbags

ViewBag.ModuleDeets = string.Format("{0} - {1}", MyEditVM.MyResults.MCode, MyEditVM.MyResults.MName);

ViewBag.StudentDeets = string.Format("{0}", MyEditVM.MyResults.SFullName);

return View(MyEditVM);

}

//POST:StudentDetail/MarkingToolView/Create

[HttpPost]

public ActionResult EditMarkingView(EditVM MyEditVM)

{

if (ModelState.IsValid)

{

//Get Marking total value to update new total value

MyEditVM.MarkTotal = MyEditVM.GetTotalMark(MyEditVM.MyResults);

//update results Database

myDatabaseCalls.UpdateMarkingResults(MyEditVM);

return RedirectToAction("ResultView", new { id = MyEditVM.ModuleID, studentID = MyEditVM.StudentID });

}

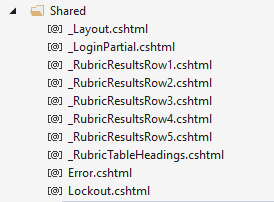
return View(MyEditVM);

}

# Troubleshoot

### Rubric Table set up

Pulling out the single values for my Marks proved to be more difficult than first realised. This proved to be an issue as well when wanting to change the cell colour of each selected mark. My solution was to create a single table for each to the Rubric Row values. This did cause a lot of repetitive code which I would like to revisit at a later stage to see if it can be simplified and update code in one place.



### MarkSet Values

At first got I couldn’t retrieved the mark set values for each of the Rubric Table Rows and had to manually create these value sets. Below was my solution. As I progressed through this project, I’ve learn how to obtain values from radio buttons and will be useful in future projects.

/// <summary>

/// Generate Marking Set for Rubric Table, returns list to use in RadioButton on MarkingToolView

/// </summary>

/// <returns></returns>

public List<MarkDTO> GetMarkSetValues1()

{

return new[]

{

new MarkDTO { MarkValue = 50},

new MarkDTO { MarkValue = 40},

new MarkDTO { MarkValue = 30},

new MarkDTO { MarkValue = 0 }

}.ToList();

}

/// <summary>

/// Generate Marking Set for Rubric Table, returns list to use in RadioButton on MarkingToolView

/// </summary>

/// <returns></returns>

public List<MarkDTO> GetMarkSetValues2()

{

return new[]

{

new MarkDTO { MarkValue = 20},

new MarkDTO { MarkValue = 16},

new MarkDTO { MarkValue = 12},

new MarkDTO { MarkValue = 0 }

}.ToList();

}

/// <summary>

/// Generate Marking Set for Rubric Table, returns list to use in RadioButton on MarkingToolView

/// </summary>

/// <returns></returns>

public List<MarkDTO> GetMarkSetValues3()

{

return new[]

{

new MarkDTO { MarkValue = 5},

new MarkDTO { MarkValue = 4},

new MarkDTO { MarkValue = 3},

new MarkDTO { MarkValue = 0 }

}.ToList();

}

### Deleting Related Data

I couldn’t delete a student due to the relation the student have with the module enrolment table. Solution was to delete all the entries in the table that contained the student FK with the student ID I had to delete.

I had to do this for each module as well to delete the Overview Titles associated with each module ID

// POST: StudentDetails/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<ActionResult> DeleteConfirmed(int id)

{

StudentDetail studentDetail = await db.StudentDetails.FindAsync(id);

//Remove Enrollment details of student using student ID

db.ModuleEnrollments.RemoveRange(db.ModuleEnrollments.Where(x => x.StudentFk == id));

db.StudentDetails.Remove(studentDetail);

await db.SaveChangesAsync();

return RedirectToAction("Index");

}