ECSE502	
Module leader:	Philip Trwoga and Artie Basukoski
Unit:	Coursework 2
Weighting:	60%
Qualifying mark:	30%
Description:	Grade Predictor
Learning Outcomes Covered in this Assignment:	Windows Forms – Design time and dynamic UI Read/Write XML for persistent storage .NET Collections OOD Development
Handed Out:	16th March 2015
Due Date:	5th May 2015 by 11:59pm
Expected deliverables	Visual Studio Solution
Method of Submission:	Electronic submission on BB via a provided link close to the submission time. The file you upload should have the following structure: Surname_IDnumber_projectName.zip for Part B YSurname_IDnumber.pdf for Part A
Type of Feedback and Due Date:	Feedback will given during student demonstration of the code in demonstration slots assigned in the Week following submission.

Grade Prediction and Performance Application

This coursework is worth 60% of the total module mark.

Set: 16th March 2015 and Due 5th May 2015

You are required to submit a zipped Visual Studio project folder via Blackboard by 11:59pm on the day

Problem Statement

You are to build a grade performance record and prediction tool for your course. This tool shall record the individual grades of a student (courseworks and exams) and can also used to predict the outcome of module marks (pass, fail, predicted mark) and also the outcome of the year and the final degree. This tool will allow an individual student to keep a record of their grades and also use the tool to predict outcomes.

User Interface

This is to be built as a .NET C# windows desktop application. You are free to design the user interface as you wish but here are some recommendations. The interface **will have a number of views**[1]:

1) A entry point to start build your course
2) Level 4 – 1st year (for setting known grades and for entering predicted grades)
3) Level 5 – 2nd year (for setting known grades and for entering predicted grades)
4) Level 6 – 3rd year (for setting known grades and for entering predicted grades)
5) Summary view – showing overall performance and predicted degree outcome (First >70%, Upper Second – 60-70%, Lower Second – 50-60%, Third – 40 – 50%, Fail <40%)
6) A view that enables the user to specify a module
7) A view that enable the user to fill in the details of a module

The view should give constant visual feedback on results. So for example, red can be used to indicate failing marks or module, green for average marks, and blue for high marks.

The years can be separated as views by using a tab control that allows you to create a number of views as a set of tabs (see Figure 2).

[1] Also see the views created in the lecture (week 10)

To give some idea of the layout, figure 3 shows a web version of the layout for level 4. Please note that you do not have to conform at all to this pattern and you are free to design the mark entry as you wish.

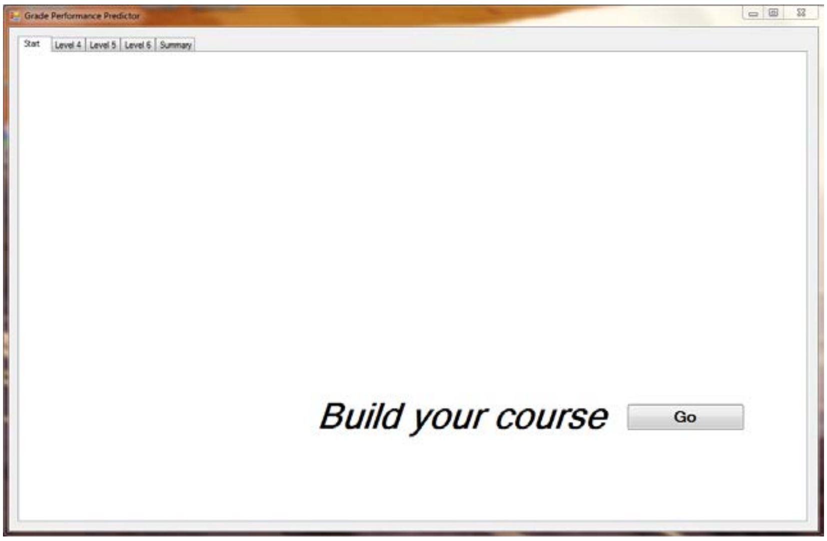
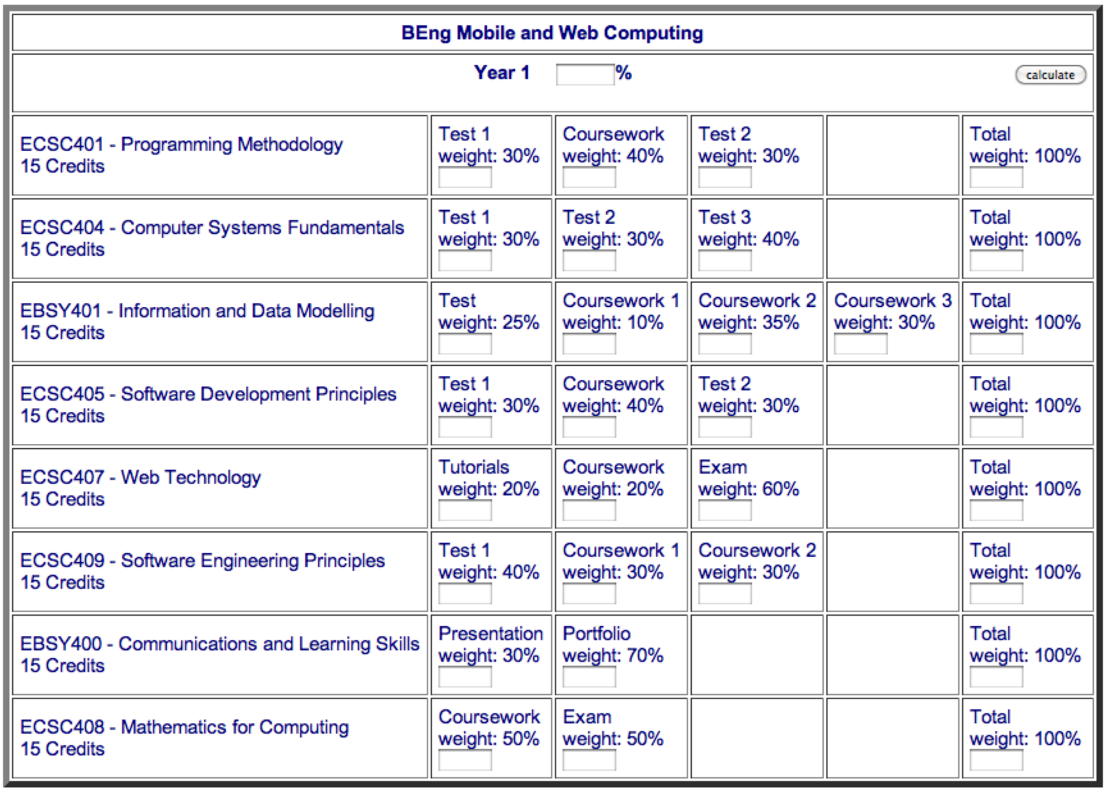


Figure 1 Tab Control shows 5 tab views (note needs a Summary tab also)



BEng Mobile and Web Computing					
Year 1 □ **%**					calculate
ECSC401 - Programming Methodology 15 Credits	Test 1 weight: 30% □	Coursework weight: 40% □	Test 2 weight: 30% □		Total weight: 100% □
ECSC404 - Computer Systems Fundamentals 15 Credits	Test 1 weight: 30% □	Test 2 weight: 30% □	Test 3 weight: 40% □		Total weight: 100% □
EBSY401 - Information and Data Modelling 15 Credits	Test weight: 25% □	Coursework 1 weight: 10% □	Coursework 2 weight: 35% □	Coursework 3 weight: 30% □	Total weight: 100% □
ECSC405 - Software Development Principles 15 Credits	Test 1 weight: 30% □	Coursework weight: 40% □	Test 2 weight: 30% □		Total weight: 100% □
ECSC407 - Web Technology 15 Credits	Tutorials weight: 20% □	Coursework weight: 20% □	Exam weight: 60% □		Total weight: 100% □
ECSC409 - Software Engineering Principles 15 Credits	Test 1 weight: 40% □	Coursework 1 weight: 30% □	Coursework 2 weight: 30% □		Total weight: 100% □
EBSY400 - Communications and Learning Skills 15 Credits	Presentation weight: 30% □	Portfolio weight: 70% □			Total weight: 100% □
ECSC408 - Mathematics for Computing 15 Credits	Coursework weight: 50% □	Exam weight: 50% □			Total weight: 100% □

Figure 2 Example layout for the level 4 tab mark entry (this is a web view but you could build something similar and how you do this is up to you.

Storage of run time data

The data for the modules (Module Code, Name, Assessment Pattern) will be created by a view that allows the specification of the module to be entered and this should be a **programmatic dynamic interface**. Once the user has finished you need to save the user data and you are free do this either as an XML file[2] or in a database of your choice (the marks awarded for each method shall be the same). When the application is run again (after closing) the system shall use the data in this file to populate the data on your interface.

Operation

1) The user elects to begin building the course by recording the name of their course
2) The user select which level they are to add a module to (so can select 4, 5, or 6)
3) The user then chooses to add a module to that level (the view should have an add + button) and selects add which should present a form (see below)
4) The new view shall appear that that allows the user to enter the title and code and the number of assessments and weightings and the credit value (15 or 30) for the module and then press a button to generate the assessment details fields
5) The user then fills out the details (Assessment name, weighting %) and submits
6) Any module entry views will now be replaced and the added module will now appear in the level summary[3] view similar to figure 3
7) This is repeated until the course is complete (120 credits per year)[4]
8) At any time the user can select the summary view (see sketch in figure 4)
9) If the application closes or if the user makes any changes while the application is running (allow the user to save manually as they may wish to make predictions but not alter the data they already have) the data shall be saved to a course XML file that records all the data
10) If the xml file exists then on launching on the application the XML files shall be used to build the interface
11) When writing XML files after changes the activity should be threaded (to simulate time taken to write date to a external database)[5]

[2] An example XML file can be downloaded from Blackboard in the Assessment area.

Note you can use this as a template and adapt it to your course (SRS is the best source for details)

[3] Note that the module details can also be deleted in case of errors etc.

[4] Note that some modules are 30 credits

[5] Some lecture notes on Threads

12) Note that the user can change the data at any time to make predictions such as 'what if I got 70% in my final project' and these should be reflected in the summary tab

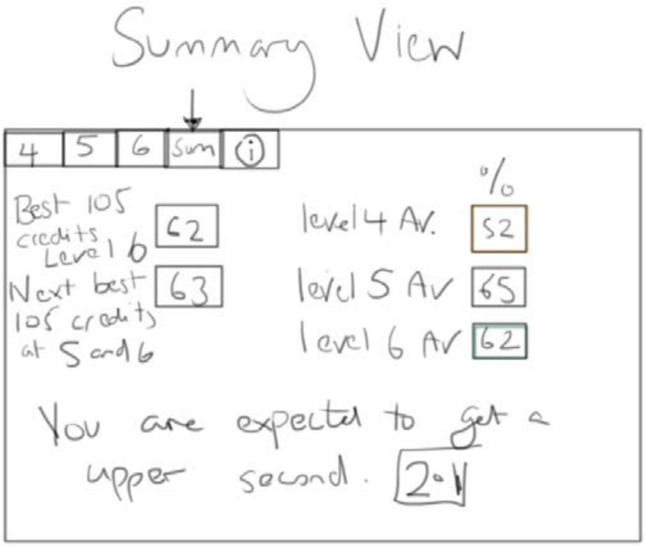


Figure 3 Sketch of a possible summary view

Deliverables:

Electronic copy to Blackboard - You are required to submit a zipped Visual Studio project folder via Blackboard by 11:59pm on the day to the Assessment area on Blackboard

Implementation

User interface (usability) – 10%
Dynamically built - Level 4,5,6 summary/predict/enter marks views – 20%
Dynamically built - create and add module view 20%
XML reader/writer/database code (writes XML only then 10% max) – 20%
Dynamically built - overall summary view (predicts final award) and prediction algorithm – 20%
Threaded file writer method – 10%

Indicative guide to marking for Part B: not done 0%, partially working up to 50%, not working but some viable code up to 30% - of maximum mark in each category) Static views (i.e. only built with the Visual Studio designer) will only get 50% of the available marks where programmatic views are required.

Notes: for top marks your interface should build dynamically and rebuild from the XML

Viva Note - IMPORTANT SO PLEASE READ

Note that the above marks are assigned with a component for defense/explanation of code and design, and any removal of marks for lack of understanding is entirely at the discretion of the marker.
Work marked without a viva is subject to a maximum mark of 35%
If an individual fails to attend the viva then they will receive a maximum of 30% for their contribution

Appendix A – self assessment form – Individual Statement

Student Name:		***Student Id:***	
Circle how much of each assessment component below you have completed. You will be asked to explain why.			
User interface (usability) – 10% (Not attempted – Some – Half – Most – Thorough – Extra) **Dynamically built - Level 4,5,6 summary/predict/enter marks views – 20%** (Not attempted – Some – Half – Most – Thorough -- Extra) **Dynamically built - create and add module view 20%** (Not attempted – Some – Half – Most – Thorough -- Extra) **XML reader/writer/database code (writes XML only then 10% max) – 20%** (Not attempted – Some – Half – Most – Thorough -- Extra) **Dynamically built summary view (predicts final award) and prediction algorithm – 20%** (Not attempted – Some – Half – Most – Thorough -- Extra) **Threaded file writer method – 10%** (Not attempted – Some – Half – Most – Thorough -- Extra)			
State what skills you gained/learnt from undertaking the project.			
State any strengths about yourself that emerged whilst undertaking the project.			
State any weaknesses about yourself that emerged whilst undertaking the project.			
State how you would do things better if you were to undertake the project again.			
Additional general or project specific comments:			
Student Signature:		***Date:***	