

COMP4: YEAR 11 DYNAMIC DATA

*Leon Singleton
Candidate No. 9178
Centre No. 23216
The Ecclesbourne
School*

Analysis

Background to the problem	1
Identification of the problem	2
Interview with Primary Client	3-5
Prospective users and Acceptable Limitations	6
Observations of the current system.....	7-9
Entity Relationship diagram of existing system	10
Data sources and destinations	11
Data volumes.....	12-13
Data flow diagrams.....	14-15
Analysis Data dictionary	16
Objectives for the proposed system	17-18
Potential solutions and justification of chosen solution.....	19-20

Design

Overall system design.....	21
Description of modular structure of system	21
Description of Record Structure's	21
Grade Predictions Model	22
Top-down Design Model	23
Database Design and Normalisation	24-26
Entity Relationship Diagram Design	27
Data Dictionary and Validation	28-30
Graphical User Interface Designs	31-49
Form Navigation Designs.....	50-51

Security and Integrity of Data	52
Identification of Storage Media	52-53
Sample of Possible SQL queries	54
Sample of Planned Algorithms and Pseudocodes.....	55-63
Testing Plan	64-68

Systems Maintenance

Form Navigation Overview.....	69-71
Database Creation	72-78
Form Overview (Technical Solution)	78-326
Completed Database Relations	327
Sample of Algorithms	328-345
Sample of SQL queries.....	346-347

Testing

System Testing.....	348-380
Navigation Testing	381-382
Secondary Interview with Primary Client.....	383
Testing Evaluation	384
Testing Evidence.....	385-499

Appraisal

Comparison of project performance against objectives.....	500-503
User Feedback	504-507
Analysis of User Feedback.....	508-510
Possible Extensions	11
Evaluations	512-513

Analysis

Project Problem Definition

Client: The Ecclesbourne School

- Mr Hewitt and Mr Nicoll (Primary)
- Teaching Staff (Secondary)

Contact:

Mr Hewitt or Mr Nicoll

Wirksworth Road,

Duffield,

Belper,

Derbyshire,

DE56 4GS,

01332 840645

RNicoll@ecclesbourne.derbyshire.sch.uk

DHewitt@ecclesbourne.derbyshire.sch.uk

Background to the problem

The Ecclesbourne School ICT department is a small department within the school that is responsible for the manipulation and storage of pupil's grade data, class data and individual student data. There are three primary users of the system, Mr Hewitt, Mr Hardy and Mr Nicoll; these will be the member of staff that I correspond with in regards to implementing a new system.

The current system in use is an Excel based spread sheet document. At the end of each academic year, Mr Hewitt and Mr Nicoll create a new Excel based spread sheet model that stores an entire assessment period, but for just one year group. Therefore, because the software in use is not bespoke, it lacks certain features and functionalities that would be desired in some areas. Data currently can be filtered and searched for in the system and a print-out of any data can be produced. Grade data can be compared against other Grade data graphically and data can be edited, inserted or deleted quite easily. The system does not currently deal with more than one year group at a time, there is also no method for predicting A2 grades based on a student's achievement records. Furthermore, the system is vulnerable because with it being an Excel document it is very easy for a user of the system to damage or modify data.

Identification of the problem

There are some noticeable flaws in the current system that the school uses to manage student's achieved GCSE grades and there are serious improvements that a new system could offer for both the clients and the school. Since the current system is Excel based it lacks the fluidity that software solution could offer so I will consider how I could use different approaches to achieving a solution.

With the current system it is also very difficult for user's to edit and add new data, because different people have different versions of the Excel based document and so any changes that a user makes will not be applied to everyone else's version of the file. A new system needs to be implemented which will allow for several user's to add and edit any data within the system and then for that data to be visible on every person's version of the system. This could perhaps be achieved by using a separate database structure which a user of the system is forced to connect to.

The current system only has very limited functionality and does not allow for grade data to be compared effectively within the system. Graphs of achieved grade data can be produced which will show the statistical representation of a subject and this can be done fairly easily using a multiple set of filters. However, the system does not effectively allow for a user to compare different grade series in the same statistical data view. With a new system this could be made possible so that data could be compared more effectively.

Another issue with the current system is that it does not do much other than act as a storage system. With a new system new features could be added such as predicting grades based on a student's achieved grades. This should be done in a way that is dynamic so that when an achieved grade is edited, it should affect the predicted grade.

Another issue I have identified is that a new system is generated on a yearly basis, this is time consuming and laborious for the school. Furthermore, it is confusing when it comes to storing several years' worth of achieved grades data. With a new system every single academic year could be stored within it; therefore the same program would be used each year and could be updated periodically to affect all stored year groups. Another great benefit of storing multiple academic years' worth of data is that they could be compared against each other to see how each year improved and to also investigate whether there has been any improvements made on a yearly basis or could help to identify underlying school performance issues. However, even by storing several year groups of data within one system it will still need to be updated on a yearly basis. As such my solution must make it easy enough for the user to bulk add records into the system. I will try to explore plausible ways to do this.

Interview with David Hewitt – The Primary Client

What are the problems with the current system?

DH: The current System is Excel based which means that it is very possible for a user to potentially damage or modify data, also at the moment any user can view any data. Another problem is that the current Excel based system only stores one assessment period of records which is a major drawback because data cannot be compared across year groups.

How much data does the current system record presently?

DH: The current system stores one assessment period for one year group which is approximately 2000 individual records.

How much data will the new solution need to be able to record?

DH: The new solution will need to allow comparison between assessment windows, so it is important that the system can store many years' worth of records.

How important is the information that is to be stored in the new system?

DH: The information that will be stored in the new system is extremely important and confidential.

Therefore, does the system require restricted or limited access to some data?

DH: Restricted access is required for the new system, so that teachers can only view certain data, for example subject specific details or class specific details. An administrator of the system (primary user) should be able to view all data.

What processes or functions can be performed by the current system?

DH: Data can be filtered, sorted very easily. Also records can be edited, deleted or inserted easily. It is also possible for a user to compare achieved grade results either graphically or using raw statistic data.

What processes or functions are to be performed by the new system?

DH: I would like the new system to be able to have all the same functions and processes as the old system but with the addition of login features, year on year comparisons and A2 predictions.

Are there any special algorithms/functions that are required for the new system?

DH: A variety of Governmental required calculations can be performed using the stored data to show school performance, student's performance or even a teacher's performance.

How regularly does data need to be inserted, updated or deleted in the new system?

DH: At the end of each Assessment window as per school assessment calendar year or more basically put at the years end.

How is Data inputted into the current system?

DH: Data is exported from an MIS and then loaded into the spread sheet.

How is data currently outputted from the system, is a hardcopy required?

DH: Dynamic querying of data using pivot tables to output searches. Hard copies of any queries can be produced with the ability to print.

What existing hardware and software requirements does the current system require?

DH: Microsoft Excel

As the end user are you willing to purchase new software or hardware resources?

DH: Yes as a department we are willing to invest in any required software or hardware that will be beneficial for the new system.

What errors and exceptions should be reported in the new system and how should they be reported?

DH: Any missing data on a system import must be flagged with an error, appropriate validation should be used when inserting or editing data within the system.

As the user, are there any specific solutions that you have in mind?

DH: A bespoke packaged software solution made using VB.NET.

Signed: David Hewitt

Summary of User interview

- I must create a system that is capable of hosting security privileges for user authentication and authorisation. My solution must be able to assign user privileges depending on the user of the system.
- My system must be capable of storing large amounts of data and must be able to process large amounts of data in an efficient way that reduces loading times for certain advanced features.
- The data that I store within my system must be kept secure because of its important nature; it should not be possible for non-authorized persons to access critical information regarding details stored within the system.
- My system must be capable of performing the features of the current system; one example of this is comparing grades graphically and allowing users to view raw statistics of data.
- I must design a solution that allows users to import data into the system solution at the end of a calendar year as easily and efficiently as possible. This should be done by importing the contents of a spread sheet.
- I must be able to produce hard copies of any important information or data outputs that can be produced via printing.
- I will be able to consider the advantages of different software and hardware resources and may be able to request their purchase for the benefits of the system.
- My user requires a bespoke solution, which means that they would like a solution that caters to their specific needs; this will mean that I will have to design specific algorithms for the system to perform and then implement them.
- My system must be able to cope with error's and non-valid data, in order to do this I will have to consider the use of different validation techniques and then implement them within my system, these validation techniques will then need to be tested effectively. Data validation and verification is extremely important because it will reduce data inconsistency within the system and may be able to eliminate it entirely.

Prospective Users and Acceptable Limitations

At present there would be lots of different users of the system. The current administrators of the system are Mr Hewitt, Mr Nicoll and the head of ICT services technician Mr Hardy with these people acting as the primary users of my system. All teaching staff within the school will act as my secondary users. All the teachers have at least some ICT skills and are computer literate, but I must be careful designing and implementing my system so as to hide the complexities of the system from the user. Ease-of use is very important because staff should be able to navigate the system without any difficulties and achieve their desired outcomes from the system with little difficulty.

Different users of the system have different needs. For example an administrator needs to be able to create new academic assessment years and import records for a new assessment year. They also need to be able to create classes and edit or insert any record along with many other admin-type functionalities. Whereas the secondary user's (teaching staff), only need to have access to individual grade records of their own classes but they should have access to features which compare grades data within the system e.g. a department's performance against another departments performance.

The limitations of the system are as follows:

- Hardware and software constraints- There is only a limited amount of hardware within the school but nevertheless each teacher at least has access to their own individual laptops. There is only a limited amount of software that I have access to on the school system but this is not a major constraint because if necessary I can arrange another meeting with one of my primary user's to discuss the advantages of new software being used for my solution. I would be able to do this since I have discovered from my initial meeting with one of my end user's that the school would be willing to purchase new software.
- My own personal Skills and Knowledge- The solution that I attempt to achieve cannot be too complex for me to solve using the resources I have available. I currently have a good understanding of programming in VB.Net and so could write a programmed solution using a number of different paradigms.
- Time Constraints- The system needs to be completed by April 29th and a user guide must accompany my implemented system for end-users to use.
- User limitations- Although every user of my system will have some basic IT ability with them all being teaching staff I can accommodate more complex features within my system without worrying too much about my potential user's. However it still remains of imperative importance that I ensure my system is as user "friendly" as possible so that anyone using my system can have a great experience.

Observations of the current system

In order to effectively analyse the existing system I was provided with a restricted copy of the existing system. This was for data protection reasons, sensitive data has been hidden in all of the following screenshots by hiding actual student's names.

Viewing students achieved grades

The picture opposite shows how you would select a student to view the grades for all their subjects. This selection can only be filtered by class. Therefore this leads to multiple students' grades being visible at once. Furthermore, a user could filter this class selection further by selecting a series of one or more subjects. The grades are formatted, and a point score is applied to each grade to show whether the student has performed better or worse than their predictions.

A	B	C	D	E	F	G	H	I	J	K	L	M	N
Student detail by class code: Random time period													Back
Class code	(All)												Use this drop down box to change student
Sum of FFT VA													
Name	subject	FFT GRADE	G	A*	A	B	C	D	E	F	G		
=A*****m	+ Drama				0								
	+ English				6								
	+ Geography	A											
	+ Maths	B			12								
	+ Music	A			0								
	+ RS	B			6								
	+ Physics	A			0								
	+ Biology	A			0								
	+ Chemistry				0								
	+ English Lit	B			6								
	+ Art	B			0								
	+ English				6								
	+ Food	B			0								
	+ French	B			0								
	+ Maths	B			0								
	+ PE	B			0								
	+ Science	B			0								
	+ H&S	B			0								
	+ English Lit	B			6								
	+ English				0								
	+ French	B			6								
	+ History	A			0								
	+ Maths	A			6								
	+ Music	A			0								
	+ ICT	A			0								
	+ Physics	A			0								
	+ Biology	A			0								
	+ Chemistry				0								
	+ English Lit	A			0								
	+ English				0								
	+ Geography	A			6								
	+ Maths	A			6								

In the existing system a user can also filter a students achieved GCSE grades by subject on a separate form. This is slightly different to the filter selection used above because this one works for all classes. On this form a user seems to be able to print the grid selection by clicking a button icon whereas on the other form a user cannot, this does not seem very consistent.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
B3 Student detail by class code: Random time period														Back
3	subject	H&S												Print
4														
5	Sum of FFT VA													
6	Class code	Name	FFT GRADE	G	A	B	C	D	E					
7	=11R/Hc1	A*****n	B											
8		B*****n	C											
9		C*****e	B											
10			D											
11		G*****a	B											
12		G*****n	C											
13		J*****e	D											
14		M*****a	B											
15		M*****e	A											
16		P*****a	C											
17		R*****a	C											
18		R*****e	B											
19		S*****e	C											
20		S*****r	B											
21		S*****y	C											
22		T*****n	A											
23		W*****y	B											

In the current system it is difficult to see which students are in which classes and it is not clear which teacher teaches a specific class. I also found that if I edited the class code of a class, it would not update across the whole system which was not very efficient.

Viewing statistics of grades

Grades by class:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	A*-C by subject and class code: Random time period														
2	Back														
3	subject Maths														
4															
6	Subject	A*	A	B	C	D	E	F	G						
7	11M8/Ma					14%	86%								
8	11M7/Ma				7%	73%	20%								
9	11M6/Ma			38%	57%	5%									
10	11M5/Ma			52%	44%	4%									
11	11M2/Ma	68%	32%												
12	11M1/Ma	90%	10%												
13	11M4B/Ma			48%	52%										
14	11M4A/Ma		52%	44%	4%										
15	11M3/Ma		67%	33%											
16	Grand Total	23%	22%	26%	18%	7%	4%	0%	0%						
17															89%

The above spread sheet shows an example of how a user would view the raw statistics of students achieving grades within a class. This seems quite efficient because a user can select a subject and then all the classes that were in that subject will dynamically appear in a grid with all the percentages of students achieving grades. Furthermore a separate dynamic table is also generated which shows the percentage of students within a class that achieved grades A*-C. One issue with this statistic view is that it does not show how many students were in each class so it is potentially flawed when it comes to comparing statistics because these classes will have different numbers of students within them. A better solution would be able provide the number of students that achieved a specific grade out of the total number of students within the class.

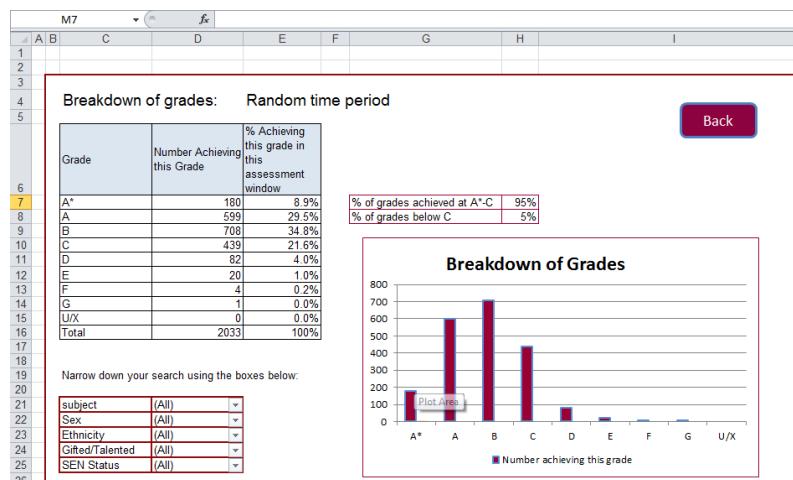
Grades by subject:

The “grade by subject” sheet is a good way to compare subject’s performance for the year. The current system deals with this particularly well. The entire list of subjects is presented when the sheet loads with all the statistic values for each subject completed. This included the percentage of students achieving a grade within a subject. Furthermore, there is an A*-C table which is completed for each subject. More interestingly on this sheet the number of students that had taken the subject is represented here. It would be nice for this information to be consistent across the system. However, there is still no data provided about the number of students achieving grades. Currently this would have to be calculated manually by the user.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
	breakdown of grades: Random time period																
	subject	A*	A	B	C	D	E	F	G								
1	art		2%	44%	34%	20%											
2	rama		24%	50%	19%	7%											
3	nglish		6%	31%	37%	23%	2%										
4	ood		31%	37%	17%	14%											
5	rench		4%	21%	26%	43%	6%										
6	eography		13%	30%	40%	14%	2%										
7	erman		35%	48%	16%												
8	raphics		20%	24%	35%	9%	7%	7%									
9	istory		12%	26%	38%	22%	1%										
10	aths		23%	22%	26%	18%	7%	4%									
11	usic		13%	47%	33%	7%											
12	as Mat		52%	33%	7%	4%	4%										
13	panish		33%	40%	27%												
14	extiles		35%	53%	12%												
15	T		5%	29%	43%	24%											
16	engineering			8%	77%	8%	8%										
17	S		12%	45%	12%	22%	4%	4%									
18	usiness		9%	36%	40%	16%											
19	E		11%	13%	28%	36%	13%										
20	S Core		4%	11%	52%	30%	4%										
21	SL		21%	27%	40%	7%											
22	ysics		7%	42%	43%	8%											
23	iology		14%	38%	41%	6%											
24	hemistry		11%	41%	41%	7%											
25	S Add		4%	25%	54%	16%	2%										
26	tec Science			25%	75%												
27	AS			18%	29%	35%	12%	6%									
28	nglsh Lit		6%	33%	35%	23%	2%										
29	us Coms		20%	37%	37%	6%											
30	omputing			12%	38%	50%											
31	rand Total			9%	29%	35%	22%	4%	1%	0%	0%						
32																	

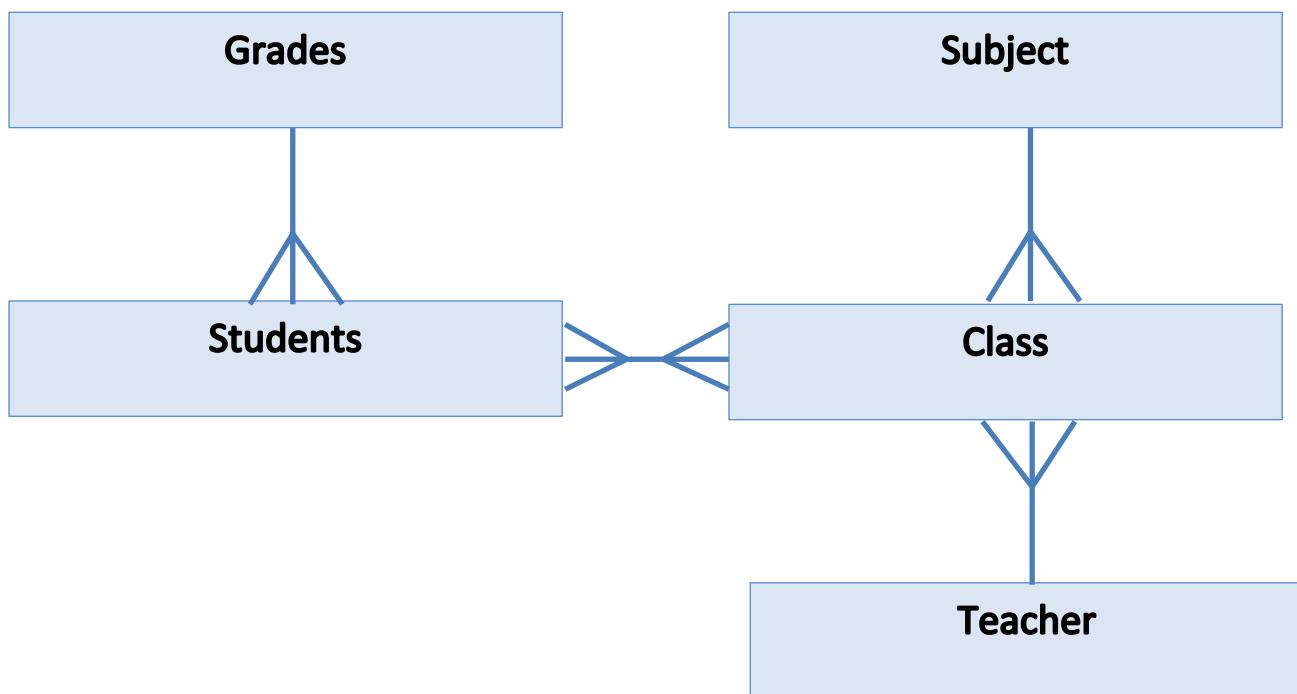
Graphically presenting grades:

Within the current system there is a very good way to present grade statistics graphically. And this can be done by a set of cumulative filters which include Subject, sex, ethnicity, gifted/talented and SEN status. This works very well and when the filters are changed the graph immediately respond to these changes along with all the statistics. Useful statistics have also been added “% of grades achieved A*-C” and “% of grades below C”. Furthermore this sheet does display the percentage of students achieving a grade alongside the number of students that actually achieved that grade. The only criticism of this sheet view is that it is not possible to compare more than one series at once.



Entity relationship diagram of the existing system

One-One relationship	
One-many relationship or many-one relationship	
Many to many relationship	



Data sources and destinations

In the existing system

What is it?	The data source	Destination
Student Details	Imported from SIMS (schools admin system)	Stored in the Excel database
Achieved Grades	Student Exam results at the end of an academic year – entered manually by an administrator of the system	Stored in the Excel database Can be filtered and then printed
Class details	Imported from SIMS (schools admin system)	Stored in the excel database, can be used to filter students grade selections
Teacher Details	Imported from SIMS (schools admin system)	Stored in the excel database
Subjects	A list of subjects is generated each year depending on the grades that have been achieved within subjects	Stored in the excel database, can be used to filter students grade selections

In the proposed system

What is it?	The data source	Destination
Student Details	These details will be imported from a report file that will contain all students' details for a specific academic year.	Stored in a Database or a Database server, will be verified on system import for inconsistencies
Achieved Grades	Student Exam results at the end of an academic year – imported by an administrator of the system	Stored in a database which the system will be able to access and filter in many different ways, printing will be an option
Class details	Imported from SIMS (schools admin system)	Stored in a Database or a Database server, will be able to identify students in classes and use this as a filter
Teacher Details	Imported from SIMS (schools admin system)	Stored in a Database or a Database server
Subjects	A list of subjects is generated each year depending on the grades that have been achieved within subjects, will also contain subjects that are taught at A2 for prediction purposes	Stored in a Database or a Database server, will be able to use subject details for filtering purposes
Login Details	An online password randomizer will be used to create a set of unique passwords depending on the number of user's	Stored in a Database or a Database server
Departments	Department details will be stored, so that the system can identify which subjects are a part of which department	Stored in a Database or a Database server, will be able to use department details for filtering purposes
Grade Type	The Grade types Achieved and predicted will be used in the system	Stored in a Database or a Database server

Data volumes

One important area to consider when I come to implementation is the amount of data I will need to store within my solution. Therefore, I must analyse the current system to see how much data it currently stores. The new system needs to be capable of storing the same amount of records as the current system with the capability for expansion in the future. I also need to take into account that the new system will store several years' worth of data and will need to be updated periodically to allow for this. Initially I will implement a system that deals with three years' worth of data to allow for extensive testing. Below I have outlined the different records stored in the current system and the number of records I will need to store in the new system.

The Data	In the current system	In the proposed system
Students Details	In the current system there are just over 200 student records stored, the system will never have to deal with roughly more than 250 students.	The new system will store multiple years' worth of data, initially 3 years' worth (~600 records). There will need to be the capability of importing records on a yearly basis.
Achieved grades	There are currently around 2000 grades stored.	Initially the system will have roughly 6000 records but will need room for expansion.
Class Details	There are currently around 60 classes stored.	Initially the system will have roughly 180 records but will need room for expansion. Approximately an extra 60 classes per year.
Teacher Details	In the school there are about 40 teachers so there are roughly 40 teacher records.	My system will have to deal with about 40 teachers; they must be able to teach classes on a year on year basis.
Subjects	There are a total of 24 subjects stored in the system.	There will need to be 26 subjects stored in the system initially, 2 of which are A2 subjects. There must be expansion available for new subjects.
Login Details	N/A	Each teacher will be assigned a unique login username and login password, so there will be as many login records as there are teacher records.
Departments	N/A	There will be a total of 8 departments in the system; this reflects the number of departments within the school.
Grade Type	N/A	Two different grade types will be stored; these will be "Achieved" and "A2 Predicted".
Academic Year	N/A	The new system will initially store three consecutive academic years but will need to cater for expansion.

Students: A student record will need to consist of first name, surname, gender, FSM (free school meals), SNS (special needs status), ethnicity, attendance, data of birth, form and year.

Teachers: A teacher record will need to consist of teacher first name and surname.

User logins: A user login will need to consist of Login username, login password, and admin permissions.

Classes: A class record will need to consist of subject, teacher, class name and academic year.

Grades: A grade record will need to consist of student, subject, grade type, grade and academic year.

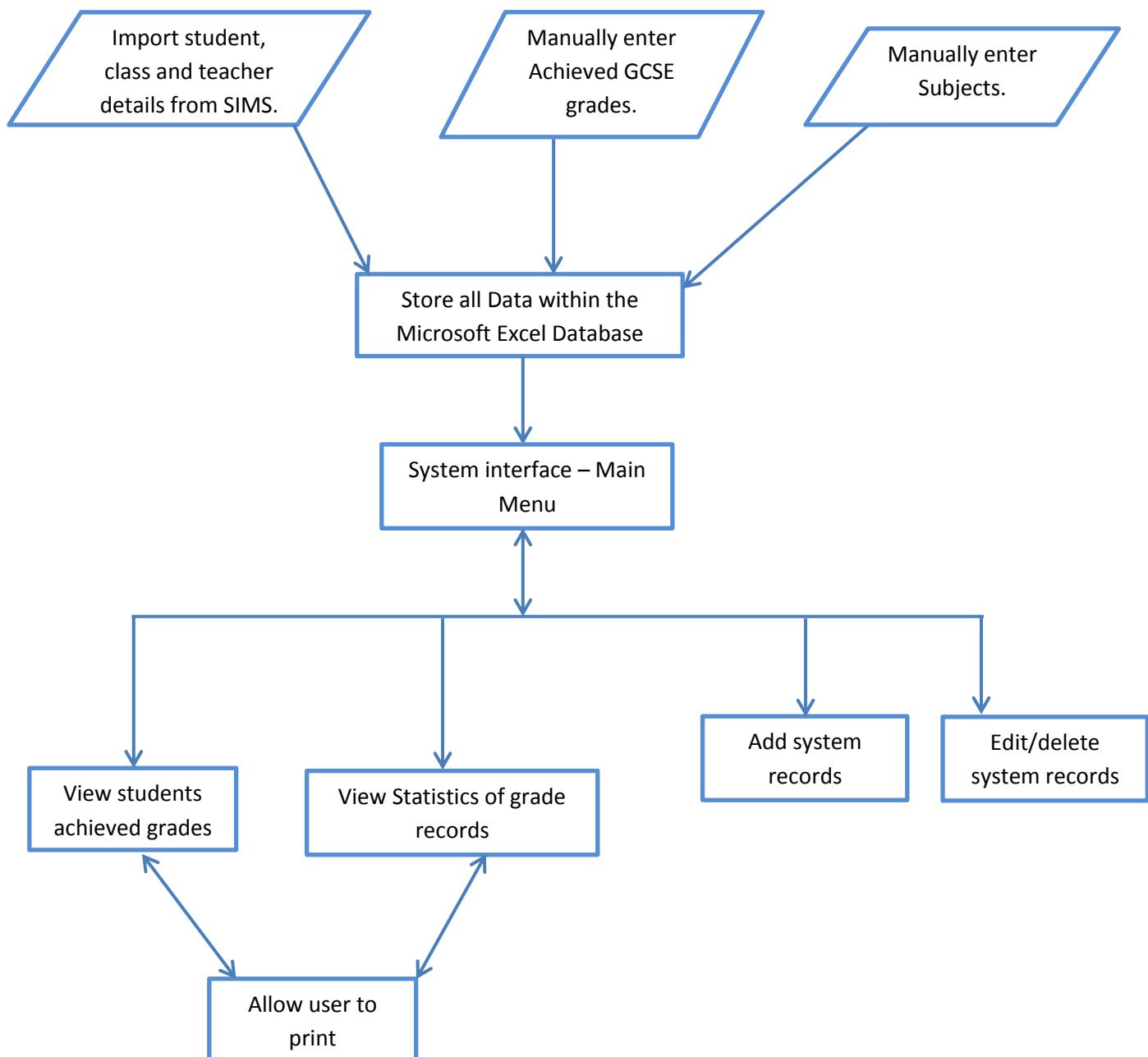
Subjects: a subject record will need to consist of subject and department.

Grade type: A grade type record will need to consist of the grade type only.

Academic Year: An academic year record will need to consist of the Academic year only.

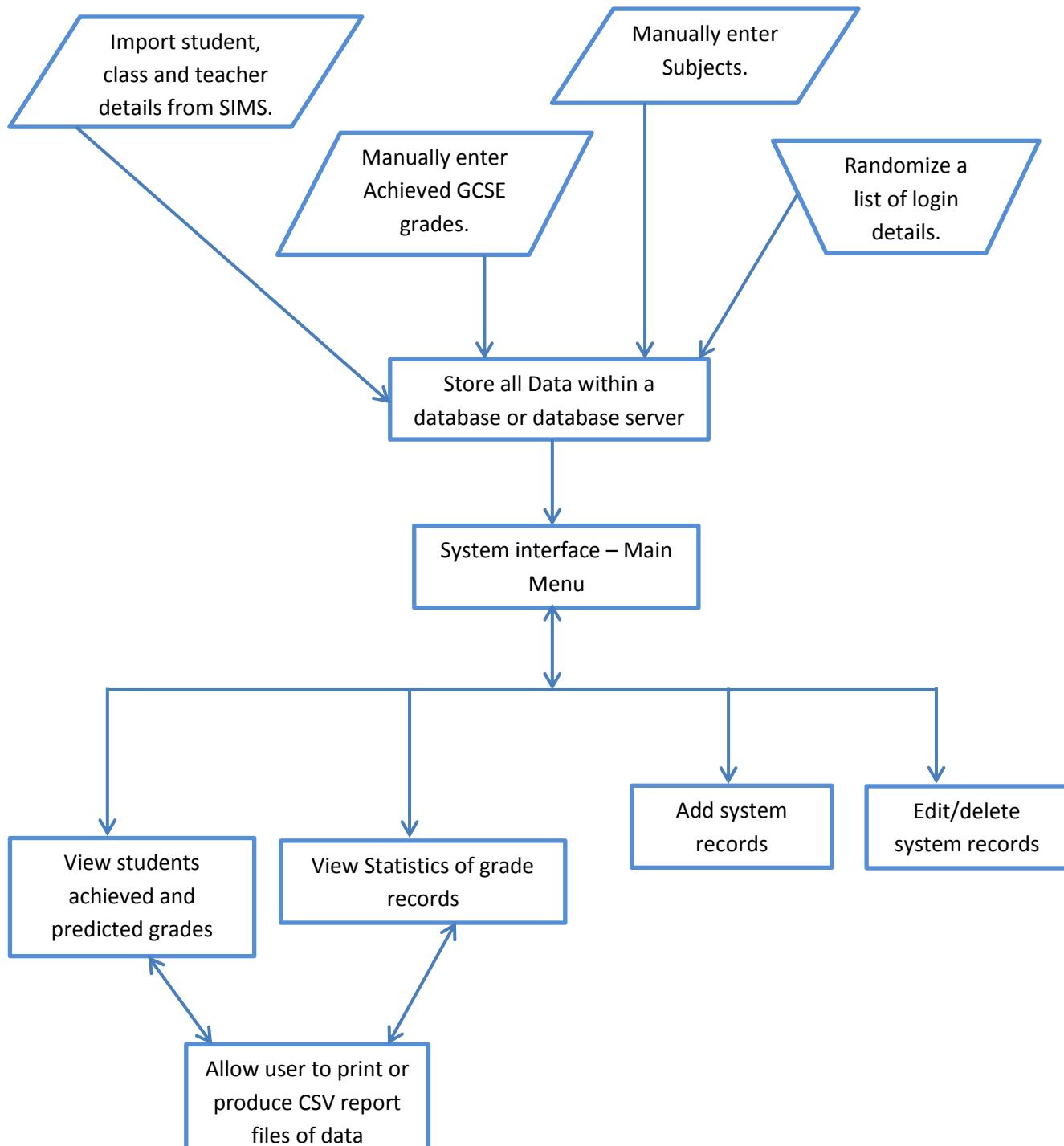
Data flow diagram (DFD) of existing system

This data flow diagram illustrates the stages of use of the current system, first of all the necessary data is all stored within the system. Once all the data has been added to the system a user can perform several processes that I have categorized within this flow diagram, this is essentially a closed loop (importing data is a one-time process).



Data flow diagram (DFD) of proposed system

The data flow diagram of the proposed system is similar to the one of the existing system, this is because all the processes within the system will be quite similar except that there will be extended functionality and more specific processes involved.



Analysis Data Dictionary

Name	Alias	Description	Data Type	Size/range
Student Forename	Stored Record	Stores the forename of a student used for identification.	String	Up to 50 characters
Student Surname	Stored Record	Stores the surname of a student used for identification.	String	Up to 50 characters
Free school meals status	Stored Record	Stores whether a student is entitled to free school meals.	Boolean	1 character
Special needs status	Stored Record	Stores information regarding any special needs a student may have, characterized by a coding schema.	String	1 character
Student's form group	Stored Record	Stores a student's form group	String	Up to 5 characters
Student Gender	Stored Record	Stores a student's group	String	Up to 1 character
Classes (names)	Stored Record	Stores the name of a class and is used as its identifier	String	Up to 30 characters
Subject	Stored Record	Stores a subject that can be used to filter.	String	Up to 30 characters
Teacher Forename	Stored Record	Stores the forename of a Teacher used for identification.	String	Up to 50 characters
Teacher Surname	Stored Record	Stores the surname of a Teacher used for identification.	String	Up to 50 characters
Various filtering selections	System combo box list	There are lots of filtering boxes within the system that allow a user to narrow a search using specific filters, so that returned results match the user's filters.	String	Various lengths, depending on what is being filtered
Various error messages	An error message	Within the system there is some validation used and errors regarding these validations are flagged.	String	Various lengths, depending on the error
Various Graphs	Graph	There is an option in the system to graphically display a single set of grades, therefore a dynamic graph is used.	N/A	N/A

Objectives for the Proposed System

From my analysis and observations of the existing system I have identified many issues with the current system and also many useful features in the current system. Also following the interview with one of my end-users I have been asked to implement a number of additional features. The proposed objectives for my system are all SMART, meaning that they are specific, measurable, attainable, relevant and time-solvable. All objectives for my system will be outlined below:

1. The system must be able to store data records for student details, class details, grade details and teacher details.
2. Student details must be able to be filtered and then an administrator must be able to select a student record for editing or deleting.
3. An administrator must be able to import large amounts of student details into the system relatively easily.
4. Records concerning the details of teacher's should be stored and an administrator should be able to edit an existing teacher's record or add a new teacher to the system.
5. Class records should be stored within the system and should be identified by their department and the subject the class is concerned with along with the teacher that teaches the class.
6. Administrators should be able to add new classes to the system and edit existing classes within the system.
7. An administrator should be able to view all of a student's classes and add or remove students from classes.
8. A login screen should be implemented which differentiates between admin user's and standard teacher user's.
9. There should be a separate viewing mode for admin user's and non-admin users.
10. An admin user should be able to use important features such as editing logins, adding new subjects to the system or new academic years to the system.
11. An administrator should be able to filter any grade within the system.
12. An administrator should be able to edit any existing grade within the system and insert any new grade for a selected student into the system.
13. Grade statistics of individual subject's performance's for achieved or predicted grades should be viewed graphically along with their raw data statistics.
14. Any user should be able to view predicted grades and filter this via student, year, class or form group.
15. Predicted grade data should be able to be exported from the system.
16. A user should be able to logout from the system.
17. A user should be able to produce hard-copies of important data binding forms.
18. Any user should be able to compare several series of data based on subject, class or department and this should be done for both achieved grades stored within the system and predicted grades.
19. There must be a model that deals with generating predicted grades.
20. A non-admin user (teacher) should be able to edit grades of students only within their own classes.
21. A non-admin user (teacher) should be able to see a student's grade that is within their class to see how they may be performing elsewhere.

22. Be able to represent grades in a format that shows the number of grades that were between A* and C.
23. The system must be user friendly and easy to use.

Potential solutions and Justification of Chosen Solution

Potential solutions:

1. A Manual paper-based system, which would allow teacher's to easily write new record's as they could simply word-process them using an existing report styled template generated by the system. An administrator of the system could write each individual grade on a student's record sheet and record details such as what subject the grade is in and what class the student was in when they achieved the grade. Multiple year groups of data could definitely be stored using this method.
 - This solution is definitely not feasible in terms of development because of time constraints, this system would take a huge amount of time to manually write record sheets.
 - This type of system is limited by physical storage space available and because of a huge volume of individual records it would take large amounts of time to search through all the records to find an individual student's record sheet.
 - A manual system would not be secure enough to store student's grades data as the only security would be to lock up certain records.
 - This type of system would not fit the requirements and the objectives that have been laid out because it would be extremely difficult to analyse data as it cannot be sorted by more than one criterion very easily and would be nearly impossible to compare large volumes of data at once.
 - One of the most important features of my system is that it is Dynamic; for example if an achieved grade is changed within the system then this should affect the individual student's predicted grades to reflect the change of the achieved grade. Therefore, with this type of system by changing one record many other records may also need to be changed. This would lead to huge scope in user-error of the system.
2. Use current off the shelf package solutions that can deal with storing huge amounts of data and manipulating data effectively. This type of system approach would allow me to implement a system using software that is readily available and is capable of dealing with such dynamic data systems. This type of approach is the one already in use by the school and they have chosen Microsoft Excel as a project environment in which to store their system.
 - This type of solution does not meet the end user's needs because any type of existing packaged solutions such as Excel will not have all the functionalities necessary for the required system whereas a bespoke solution which I intent to offer will.
 - Common drawbacks with these types of system are the security and integrity of data, in most cases it is hard to restrict different types of user access to the system and this was one of the problems my meeting with Mr Hewitt unearthed with the current Excel based system.
 - This type of solution is far too similar to the abilities of the current system and although it would more than possible to achieve in regards to time constraints and user knowledge it would be nothing more than an improvement of the old system rather than a drastic upgrade which is required.

3. A fully computerised and part web-based system which would eliminate the need for data to be stored using Excel. Teachers and admins would be able to login to the system using the schools interactive learning portal and view records there. Records with this type of system would be stored in a secure database system such as Microsoft Access.
- This solution is not feasible in terms of my abilities it would be far too difficult to develop a web based system with the time constraints I have. It would also require me having access to the school's username and password data which would compromise the whole school's system security.
 - The end user wanted a complete software packaged solution that performs all the features of the old system and the new features which have been outlined in the specific objectives of the project.
 - However a vast advantage of this system is that teaching staff would not have to adapt to a new working environment because they will most likely be able to use the schools learning portal services confidently and so this would facilitate all of the client's initially outlined needs of a "user-friendly" system.

Chosen Solution:

4. A fully computerised system which is capable of meeting all of the end-user's objectives and requirements and will eliminate all time-consuming processes within the current system and limitations of the current system. Data would be stored in a normalised database relation either in a SQL server or Microsoft Access. Data would then be accessed by the packaged software solution using SQL filtering queries and updated, edited and inserted in the same manner.
- This is the most feasible solution because I could use Microsoft Visual Studio as a programming environment and using my current coding abilities of VB.Net to create a solution that is capable of meeting all the user's needs and requirements as outlined by the objectives for the proposed system. Furthermore, this type of system would be easily manageable as all of my primary user's (administrators) have extensive knowledge of programming in VB.Net.
 - The teaching staff should not have trouble using this type of system since they are all computer literate and I will be able to make the interface of my solution very "user-friendly".
 - There would be no repeating data in the system because a fully normalised database approach would be implemented.
 - The system would be dynamic, for example if two users were both using the software at the same time and one user updated a record the other user would also be able to see that updated record in their use of the system.

Design

Overall System Design

In order to design my system effectively I will plan extremely important features such as the HCI interface (GUI), the storage of Data, algorithms that will be implemented Validation features. As outlined in my analysis section I plan to use Microsoft Visual Studio as a means to implement my solution, I will use VB.Net as my chosen programming language because of my prior knowledge of using the language. I will also be using Macromedia Fireworks to design system icons and graphics for use within the system. Furthermore, I will be using Microsoft SQL server studio to store my database which will contain all records for my system.

Description of Modular Structure of System

Within my system I will have a module that allows me to re-use common routines that will be used several times within my system. This allows the footprint of my programmed solution to be smaller, but most importantly it allows future management of the system to be more easily done. For example if a new feature needs to be added to the system then or an existing feature needs changing, only one section will need to be changed rather than several sections of essentially the same code. At present I believe that the connection to my SQL database structure will be done via a module routine and the prediction of grades will also be done via module routine. In order for these to work I need to make my system robust enough to deal with advanced cases of parameter passing. Furthermore, my code will consist of separate sub-routines and functions which will allow me make my code more usable, these functions and sub-routines can then be called within other sub-routines. Some of my sub-routines will be event driven, this means that they will perform their encasing actions when the event change has been met, an example of this in sue is when a user selects an item from a comb-box filter selection.

My system will also be modulated in other areas such as form navigation design. Rather than having a large amount of forms I will attempt to use my forms in a clever way that allows me to limit the number of forms in my system by either making certain objects on forms visible or non-visible depending on the functionality that the user has requested.

Description of Record Structure's

In order to import or export data to and from the system I will be using CSV file formats, this is because in order to move tabular data from one source to another between programs that natively operate on incompatible formats, data values have to be separated by commas. In order for data to be imported from a source such as excel which stores data in a non-proprietary format I will need to import it as a CSV file and then use a delimiter to store the contents of the CSV file as an array which I can use to be imported to my database. On the other hand in order for data to be exported from my system, it can easily be done in CSV form and then can be imported by a spread sheet program such as Microsoft Excel.

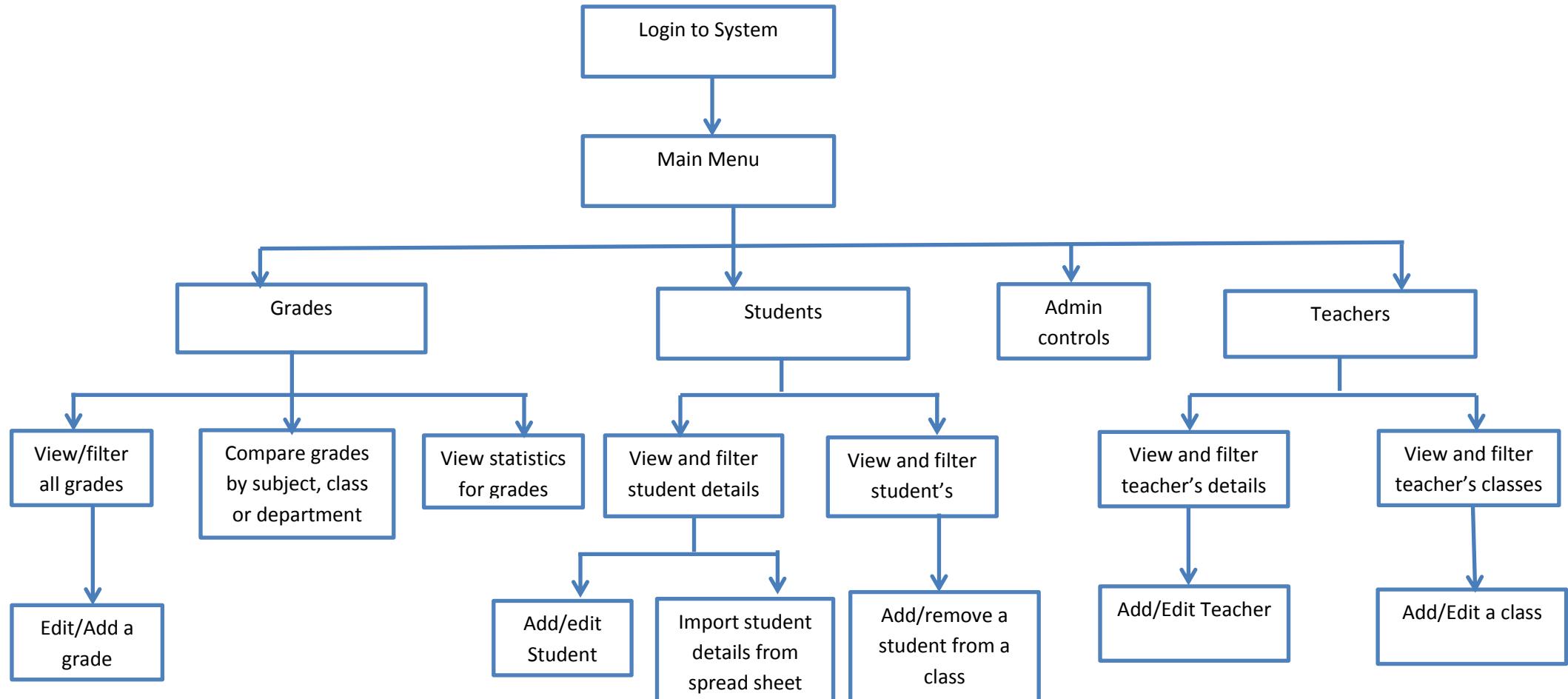
In regards to storing data that the system will use I will use Microsoft SQL server to store all of my data within a set of normalised table relations. In order to use this stored data I will use SQL queries within my Visual Basic solution to manipulate data that will be stored in the database.

Grade Predictions Model

In order to implement grade predictions into my system it is of high importance that I have a grade predictions model which deals with subject weightings for both GCSE and A2. In the tables below each subject is given a rating which is based on the perceived relative difficulty of the subject. This weighting is then in the case of a GCSE predicted grade used towards a mean points score. In the case of predicting A2 I will use the mean average point score to generate a predicted grade for a given subject based on its perceived relative difficulty at A2 level. These weightings have been given based on Governmental ratings of perceived subject difficulties after I have carried out some research, however the weightings I have given do not at all reflect the actual Governmental ratings of subjects but are used in accordance with them to apply a realistic weighting value.

<u>Subject</u>	<u>GCSE Weighting</u>	<u>A2 weighting</u>
Art	0.93	1.07
Business Studies	0.94	1.06
Drama	0.86	1.14
English Language	0.98	1.02
English Literature	0.98	1.02
Food Technology	0.88	1.12
Geography	1	1
Health And Social care	0.9	1.1
History	1.04	0.94
ICT	1	1
Leisure and Tourism	0.87	1.13
Maths	1.11	0.89
French	1.12	0.88
Spanish	1.12	0.88
German	1.12	0.88
Graphics	0.92	1.08
Music	0.96	1.04
Religious Studies	0.95	1.05
Resistant Materials	0.89	1.11
Physical Education	0.88	1.12
Computing	1.08	0.92
Biology	1.18	0.82
Chemistry	1.22	0.78
Physics	1.2	0.8
Textiles	0.9	1.1

Top-down Design Model



Database Design and Normalisation

Since I will be using a database for storing data within my system it is important that it is normalised and in third normal form to avoid data inconsistencies and the duplication of data. By doing this I will in turn save space and eliminate all non-atomic data from the system. I will also have no non-key partial dependencies.

In order to indicate which fields are primary keys or where there is more than one primary key (composite key) I have highlighted these fields in **bold**.

The table below is the **first table without any normalisation**.

Non-normalised relations

GradeID	Subject	Department	Student Firstname	Student Surname	DOB	Gender	Academic Year	Grade	Grade Type	Level	Class Group	Teacher Firstname	Teacher Surname	Login Username	Password	Admin

The table below is also a part of the above table:

FSM	SNS	Ethnicity	Attendance	Form	Year

This table contains redundant non-atomic data. Lots of the attributes are repeated several times within this table which would result in thousands upon thousands of records if I were to store data in my system using this format.

In order to normalise this table to **first normal form** repeating attributes need to be moved to separate tables and assigned primary keys.

In order to normalise to **second normal form** the tables need to be in first normal form and contain no partial key dependencies.

In order to normalise to **third normal form** it needs to be in second normal form and have no non-key dependencies.

My fully normalised database relations (third normal form)

These tables are all in third normal form and there will be no non-atomic data if I am to store data for my system using these tables. There are no functional dependencies existing between attributes at this point of normalisation.

Student

StudentID	Firstname	Surname	Gender	FSM	SNS	Ethnicity	Attendance	DateOfBirth	Form	Year	DatasetID

Class

ClassID	SubjectID	TeacherID	ClassGroup	DatasetID

Department

DepartmentID	Department

Grade

GradeID	StudentID	SubjectID	LevelID	GradeTypeID	Grade	DatasetID

GradeType

GradeTypeID	GradeType

Level

LevelID	Level

Teacher

TeacherID	LoginID	Firstname	Surname

Teaches

TeachesID	TeacherID	ClassID

UserLogin

LoginID	LoginUsername	Password	Admin

StudentGroup

StudentGroupID	ClassID	StudentID

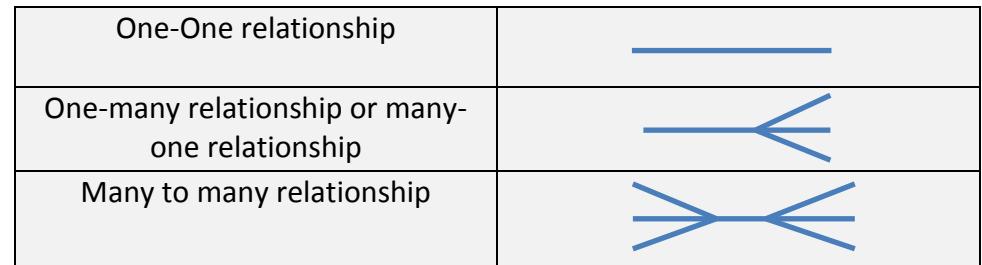
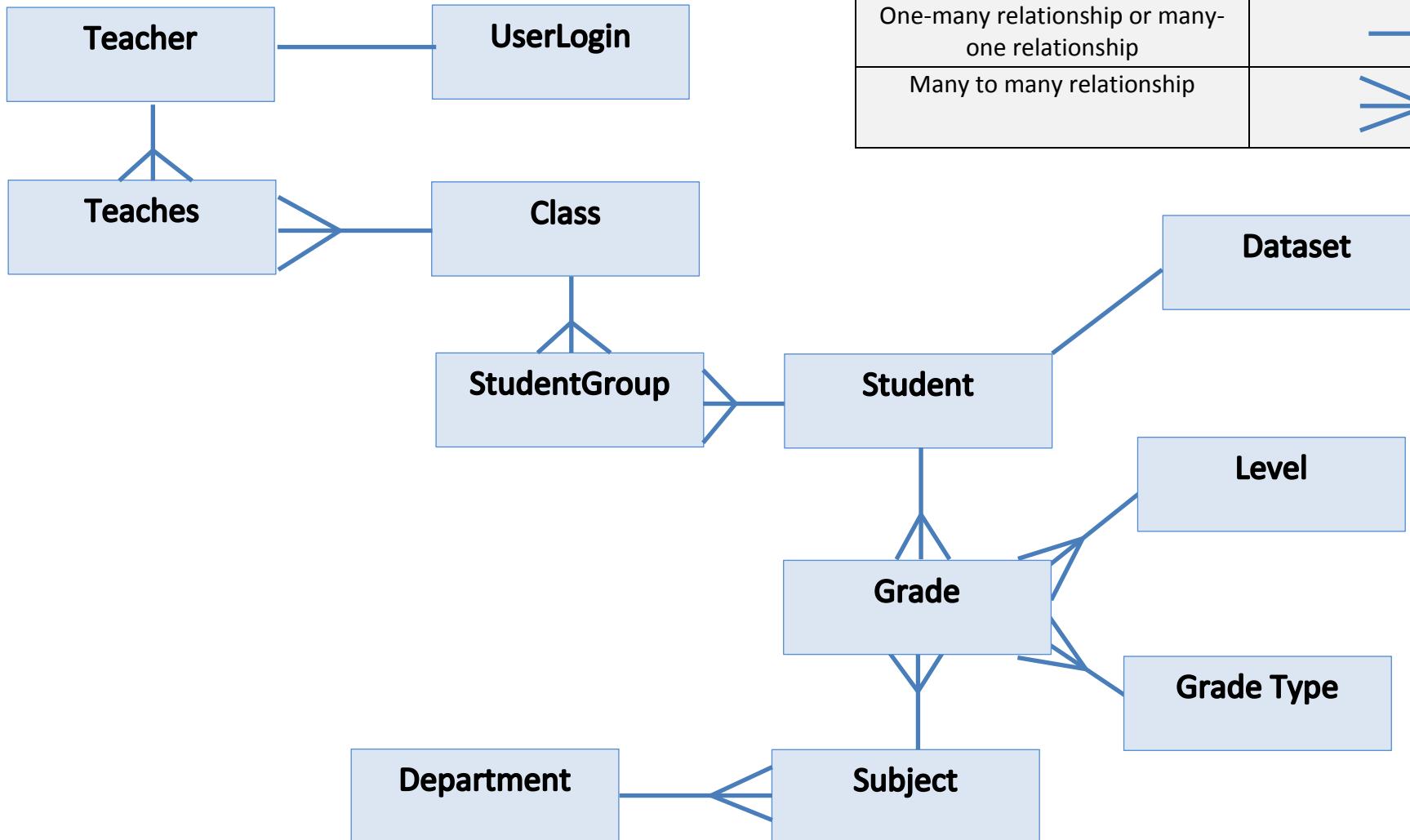
Dataset

DatasetID	Year

Subject

SubjectID	DepartmentID	Subject

Database Table (entity) Relationship Diagram Design



Data Dictionary and Validation

The following data dictionary below contains all the data types that could be inserted, edited or deleted within the system by a user, there will also be ID's that are inserted or deleted from the system but these will be done dynamically so I have not included them within the data dictionary. I have also included the examples of validation checks that I expect to be carried out on each field as well as example data that could be valid for a particular field.

Validation

In my system I will employ different validation techniques to make sure that any data that is either inserted into the system or any data that is currently in the system remains consistent. It is extremely important that data is kept consistent within the system because this will mean that anomalous records cannot exist and incorrect records cannot affect important functions such as my predictions algorithm. The main type of validation that I will use across my entire system is a presence check type of validation, this type of validation ensures that null data cannot be stored in my database. However, for effective validation presence checking cannot be used solely on its own. As such I will also use list checks which restrict a user's choice of data entry and length checks to ensure that data uses the correct amount of characters. Another type of validation I will have to use is a format check to make sure that data is entered in the right format; this would apply to a student's date of birth. Other types of validation features will most likely be used when I come to implement my system.

Field	Data Type	Length	Validation Check	Validation Description	Valid Data
Class Group	String	1-17 characters	Presence check, Length check	Allows the user to use either numbers or letters.	"Class Group 101"
Year (Academic)	Integer	1-4 characters	Presence check, Length check, List check	Restricts a user to only enter a year format record.	"2016"
Department	String	1-30 characters	Presence check, Length check, List check	Restricts the number of characters the user can enter as the name for the record. In some cases the user will be restricted to one of several choices.	"Science"
Grade	String	1-2 characters	Presence check, Length check, List check	Restricts the number of characters the user can enter as the grade for the record. In some cases the user will be restricted to one of several choices.	"A*"
Grade Type	String	1-30 characters	Presence check, Length check, List check	Restricts the number of characters the user can enter as the grade type for the record. In some cases the user will be restricted to one of several choices.	"Working Towards"
Level	String	1-5	Presence check,	Restricts the number of	"KS5"

		characters	Length check, List check	characters the user can enter as the level for the record. In some cases the user will be restricted to one of several choices.	
Firstname	String	1-30 characters	Presence check, Length check	Restricts the number of characters the user can enter as the name for the record.	"Alexander"
Surname	String	1-30 characters	Presence check, Length check	Restricts the number of characters the user can enter as the name for the record.	"Williams"
Gender	String	1 character	Presence check, Length check, List check	Restricts the number of characters the user can enter as the gender for the record. In some cases the user will be restricted to one of several choices.	"F"
FSM	String	1 character	Presence check, Length check, List check	Restricts the number of characters the user can enter as the FSM code for the record. In some cases the user will be restricted to one of several choices.	"Y"
SNS	String	1 character	Presence check, Length check, List check	Restricts the number of characters the user can enter as the SNS code for the record. In some cases the user will be restricted to one of several choices.	"P"
Ethnicity	String	1-30 characters	Presence check, Length check, List check	Restricts the number of characters the user can enter as the ethnicity for the record. In some cases the user will be restricted to one of several choices.	"Asian"
Attendance	Double	1-5 characters	Presence check, Length check	Restricts the number of characters the user can enter as the attendance for the record.	"98.7%"
DateOfBirth	Date	N/A	Presence check, Length check, Format check	Restricts the format the date of birth can be entered in.	"2000-07-19"
Form	String	1-3 characters	Presence check, Length check, List check	Restricts the number of characters the user can enter as the form for the record. In some cases the user will be restricted to one of several choices.	"12C"
Year (student)	Integer	1-2 characters	Presence check, Length check, List	Restricts the number of characters the user can enter	"11"

year			check	as the year for the record. In some cases the user will be restricted to one of several choices.	
Subject	String	1-30 characters	Presence check, Length check, List check	Restricts the number of characters the user can enter as the subject for the record. In some cases the user will be restricted to one of several choices.	"Computing"
Firstname (Teacher)	String	1-30 characters	Presence check, Length check	Restricts the number of characters the user can enter as the name for the record.	"Alexander"
Surname (Teacher)	String	1-30 characters	Presence check, Length check	Restricts the number of characters the user can enter as the name for the record.	"Williams"
LoginUsername	String	1-30 characters	Presence check, Length check	Restricts the number of characters the user can enter as the login name for the record.	"AWilliams"
Password	String	1-10 characters	Presence check, Length check	Restricts the number of characters the user can enter as the password for the record.	""
Admin	Integer	1 character	Presence check, Length check	Can either be a 1 or 0 to indicate a user's permissions status.	"1"

Graphical User Interface Designs (HCI) (Including navigation designs)

These designs below are my first outlines of what I would like my system interface to resemble. They will however most likely not represent the final designs because during implementation my designs may change. Therefore the designs below are my first prototypes of each form and will act as a starting point for me to work on regarding my implementation and later concepts. Another thing to note about these initial designs is that there may be more forms in the finished implementation than the number of forms designed here. For my prototypes I will use the online web based service called www.moqups.com in order to create my prototypes.

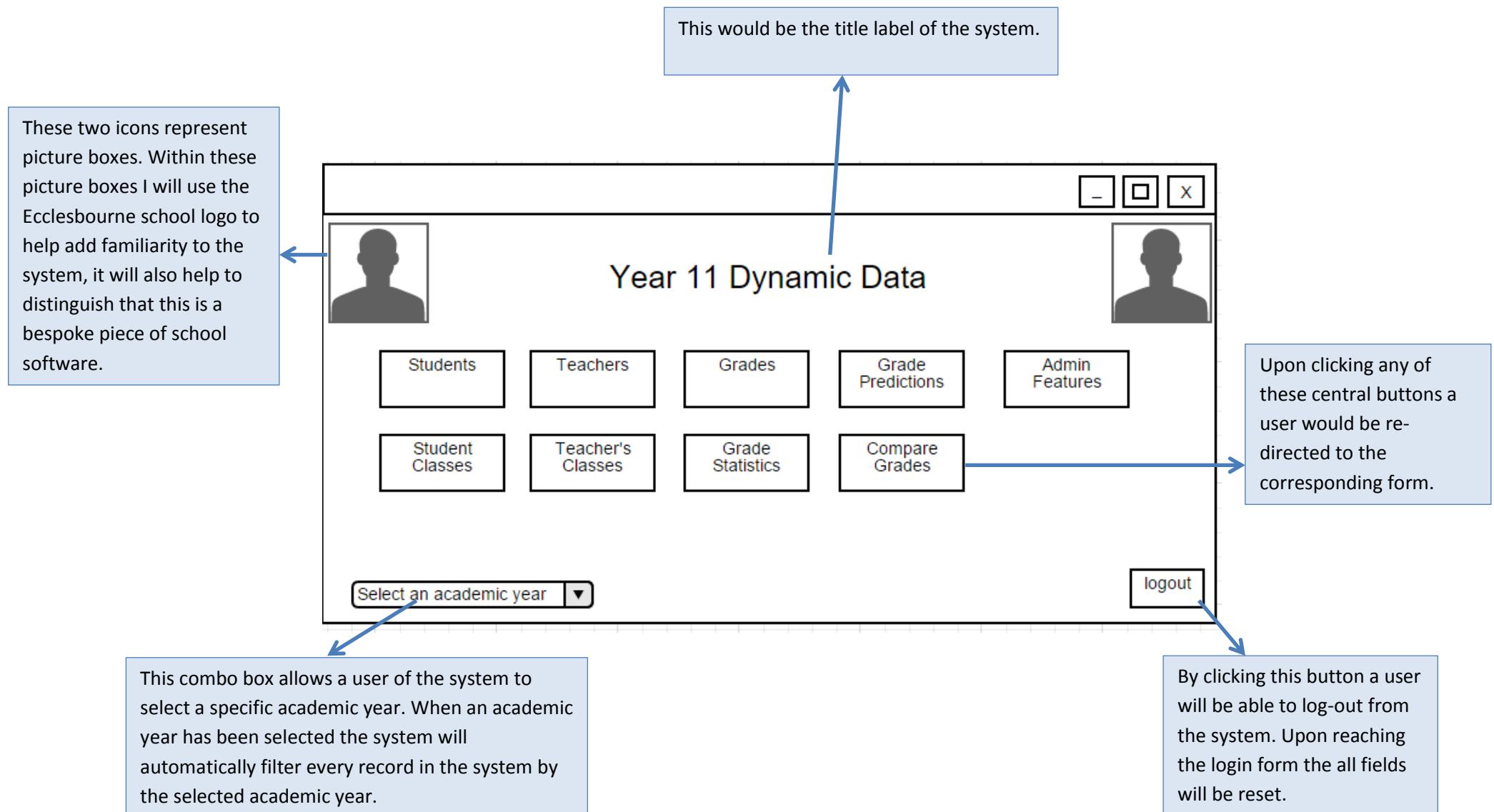
Login form

The form consists of a header with three icons, a 'Username' field with placeholder 'firstname', a 'Password' field with placeholder 'Surname', and a 'Login' button.

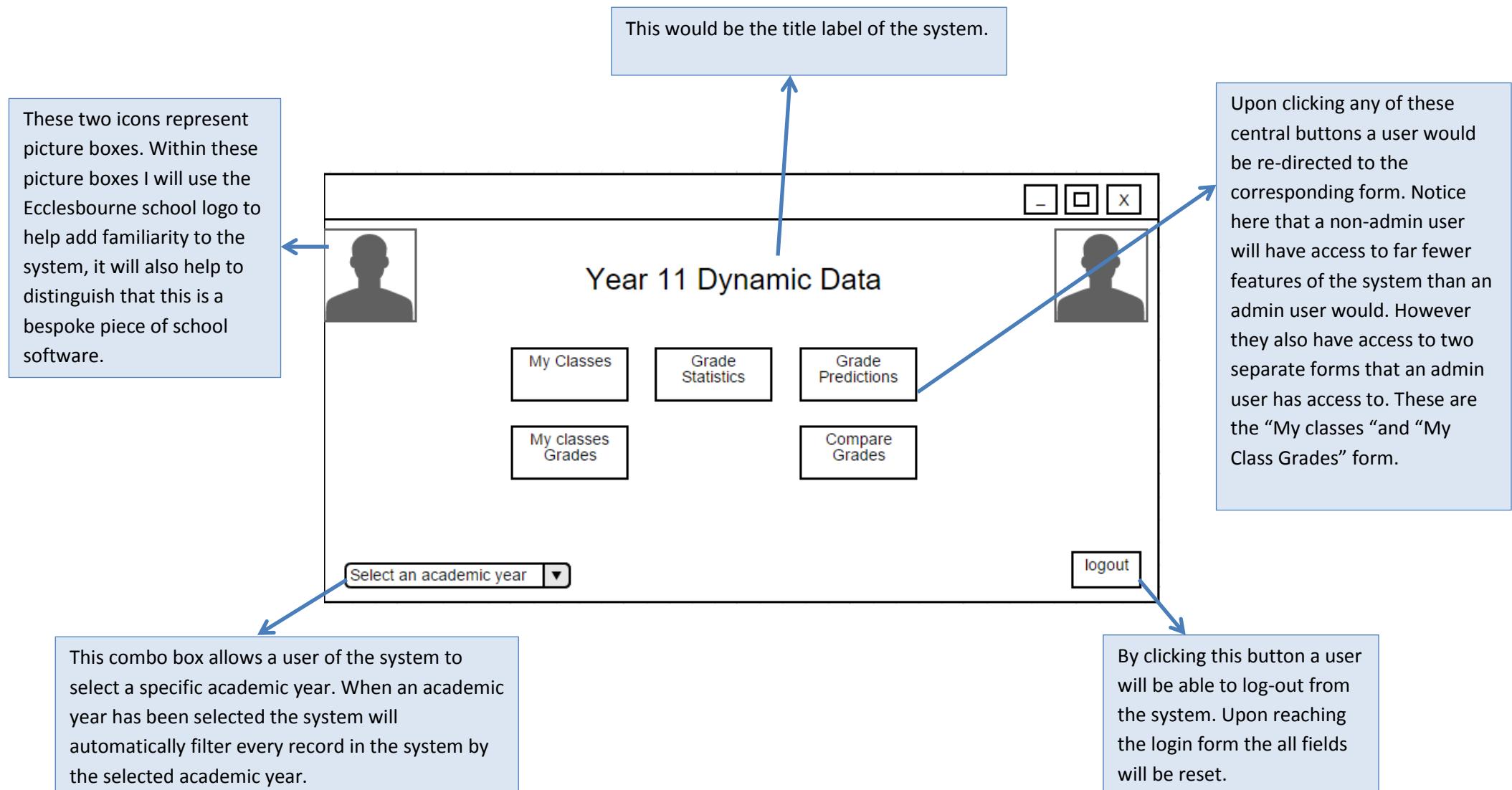
These two fields would allow a user to enter their usernames and passwords into the system. The password field would hide the user's password by using the "*" character to replace a user's typed in password.

This login button would be used by the user to login into the form, an algorithm will check whether the user is an admin or non-admin user and then apply the corresponding permissions. Login validation will also be checked when this button is clicked.

Main Menu for admin user



Main Menu for non-admin user



Student Form

This diagram illustrates the layout and functionality of the Student Form application. The form is titled "Search Student Details" and includes the following components:

- Form Header Title:** "This is the Form header title." (Top Left)
- Search Group Box:** "From this group box a user will be able to search student's using a number of different filters." (Left Side)
- Search Buttons:** "Search", "Add new Student", and "Import Student CSV".
- Return Button:** "Upon clicking this button a user would be returned to the main menu form." (Top Right)
- Data Grid:** "This data grid would display all the student records that correspond to the user's academic year choice and match the search criteria." (Bottom Right)

The data grid has three columns labeled "Head 1", "Head 2", and "Head 3". The rows are labeled "Cell 1" through "Cell 12".

▼ Head 1	▼ Head 2	▼ Head 3	
Cell 1	Cell 2	Cell 3	
Cell 4	Cell 5	Cell 6	
Cell 7	Cell 8	Cell 9	
Cell 10	Cell 11	Cell 12	

Teacher Form

This is the Form header title.

Upon clicking this button a user would be returned to the main menu form.

From this group box a user will be able to search teacher's using a number of different filters.

Upon clicking this button a user would be re-directed to the add new teacher form.

This data grid would display all the teacher records that correspond to the user's academic year choice and match the search criteria.

The form is titled "Search teacher Details". It features a "Filter by" section with "Filter" and "Where" input fields, and a "Search" button. Below this is an "Add new Teacher" button. To the right is a data grid with three columns labeled "Head 1", "Head 2", and "Head 3". The grid has four rows of data labeled "Cell 1" through "Cell 12". At the top right of the form is a "Return to Main Menu" button. The top right corner of the window has standard window controls (minimize, maximize, close).

Student Classes

This is the Form header title.

Upon clicking this button a user would be returned to the main menu form.

From this group box a user will be able to search student's using a number of different filters. Then they will be able to select an individual student and view all of their classes.

From this group box a user will be able to search classes using a number of different filters. Then they will be able to select an individual class and choose whether to add a new student to a selected class or view the current student's within a class. After viewing the current students within a class a user will be able to remove any of those students.

This data grid would display all the student class records that correspond to the user's academic year choice and match the search criteria.

Head 1	Head 2	Head 3
Cell 1	Cell 2	Cell 3
Cell 4	Cell 5	Cell 6
Cell 7	Cell 8	Cell 9
Cell 10	Cell 11	Cell 12

Teacher Classes

This is the Form header title.

Upon clicking this button a user would be returned to the main menu form.

From this group box a user will be able to search classes using a number of different filters. Then they will be able to select an individual class and edit the details of one of those classes.

This group box will allow a user add a new class to the system. To do this they will have to select a valid set of data and then press the add class button with all fields being completed.

This data grid would display all the class records that correspond to the user's academic year choice and match the search criteria.

Grades

From this group box a user will be able to filter grades using a number of different filters. The difference with this form is that filters can be used in conjunction with one another to more accurately filter grades. Then they will be able to select an individual grade and edit the details of one of those grades.

This is the Form header title.

Upon clicking this button a user would be returned to the main menu form.

▼ Head 1	▼ Head 2	▼ Head 3	
Cell 1	Cell 2	Cell 3	
Cell 4	Cell 5	Cell 6	
Cell 7	Cell 8	Cell 9	
Cell 10	Cell 11	Cell 12	

Upon clicking this button a user would be re-directed to the add grade form.

This data grid would display all the grade records within the system and match the search criteria.

Grade Statistics

This is the Form header title.

Upon clicking this button a user would be returned to the main menu form.

From this group box a user will be able to filter a grades selection using a number of different filters. Once a valid selection of grade statistics has been selected a user can then click the view statistics button which will display the statistics of grades for their chosen grade selection criteria.

This chart would display the grade statistics for the user's search selection.

The window title is "Grade Statistics". The toolbar buttons are standard window controls: minimize, maximize, and close. The "Return to Main Menu" button is located in the top right corner of the main window area. The "Filter Grade Statistics" group box contains fields for "Subject", "year", "Subject", "level", and "graph type", each with a corresponding input field. A "view statistics" button is located at the bottom of the group box. To the right of the group box is a bar chart with five bars of varying heights. The entire window is enclosed in a thick black border.

Compare Grade Statistics

From this group box a user will be able to filter a grades selection using a number of different filters. Once a valid selection of grade statistics has been selected a user can then click the view statistics button which will display the statistics of grads for their chosen grade selection criteria. With this form a user will be able to keep adding series to the graph so that they can compare them.

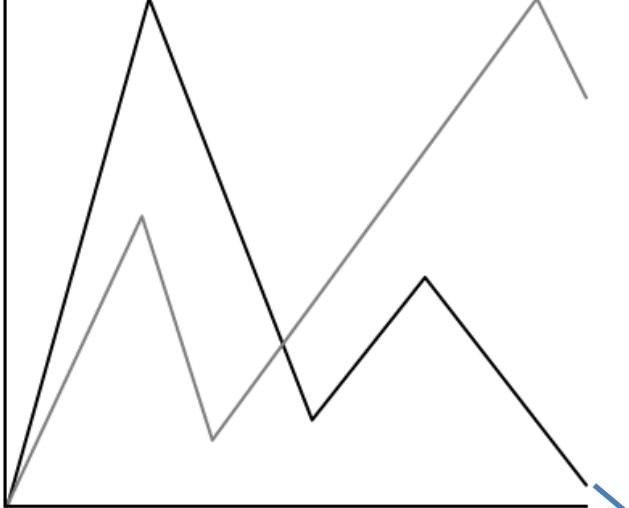
This is the Form header title.

Upon clicking this button a user would be returned to the main menu form.

Filter Grade Statistics

Subject
year
Subject
level
graph type
Add statistics series

Return to Main Menu



This chart would display the grade statistics for the user's search selection and would allow the user to compare several chart

Grade Predictions

From this selection of group boxes a user will be able to filter a dynamic predicted grades selection using a number of different filters. Once a valid selection of predicted grades has been selected a user can then click the search button which will display the dynamic predicted grades for their chosen selection criteria.

This screenshot shows a user interface for 'Grade Predictions'. On the left, there are four filter sections: 'Filter by form' (with dropdowns for 'form' and 'Search'), 'Filter by student' (with dropdowns for 'Filter' and 'Where', and a 'Search' button), 'Filter by Year' (with dropdowns for 'Year' and 'Search'), and 'Filter by Class' (with dropdowns for 'Class' and 'Search'). In the center, there is a data grid with three columns labeled 'Head 1', 'Head 2', and 'Head 3'. The grid contains 12 cells labeled Cell 1 through Cell 12. At the top right of the form are three buttons: 'Export Spreadsheet CSV', 'Return to Main Menu', and a close button. A callout box points to the 'Form header title' at the top left of the form area. Another callout box points to the 'Export Spreadsheet CSV' button, stating it triggers a save file dialog for a CSV file. A third callout box points to the 'Return to Main Menu' button, stating it returns the user to the main menu. A fourth callout box points to the data grid, stating it displays dynamically generated grade predictions based on search criteria.

This is the Form header title.

Upon clicking this button a save file dialog will appear which will allow the user to save a formatted CSV file of the current data grid.

Upon clicking this button a user would be returned to the main menu form.

From this selection of group boxes a user will be able to filter a dynamic predicted grades selection using a number of different filters. Once a valid selection of predicted grades has been selected a user can then click the search button which will display the dynamic predicted grades for their chosen selection criteria.

Upon clicking this button a user would be returned to the main menu form.

This data grid would display all the dynamically generated grade predictions that correspond to the user's choice the search criteria.

Developer Features

This is the Form header title.

Upon clicking this button a user would be returned to the main menu form.

When a user selects an admin action from the admin action combo box a group box that corresponds to that admin action will appear. Along with this a data grid will appear which will contain all the current database records that correspond to the chosen admin action. For example if the user chooses the 'add a new dataset' admin action a group box will appear allowing the user to add a new dataset, and the data grid will be filled with the current stored datasets.

The screenshot shows a Windows application window titled "Admin Features". At the top right are standard window controls (minimize, maximize, close). Below the title bar is a toolbar with three buttons. A "Return to Main Menu" button is located in the center of the toolbar. The main content area contains a "Select Admin feature" dropdown menu, a "New subject" group box with a "subject" input field and an "Add" button, and a 4x3 data grid. The data grid has columns labeled "Head 1", "Head 2", and "Head 3". The data grid rows are labeled Cell 1 through Cell 12. Arrows point from the callout boxes to specific UI elements: one arrow points to the "Select Admin feature" dropdown, another to the "Return to Main Menu" button, and a third to the data grid itself.

Head 1	Head 2	Head 3
Cell 1	Cell 2	Cell 3
Cell 4	Cell 5	Cell 6
Cell 7	Cell 8	Cell 9
Cell 10	Cell 11	Cell 12

My Classes

This is the Form header title.

Upon clicking this button a user would be returned to the main menu form.

This data grid would display all the classes that belong to the logged in teacher as soon as the form loads. From here a teacher will be able to remove students from one of their classes or view all student's within one of their classes.

▼ Head 1	▼ Head 2	▼ Head 3
Cell 1	Cell 2	Cell 3
Cell 4	Cell 5	Cell 6
Cell 7	Cell 8	Cell 9
Cell 10	Cell 11	Cell 12

My Class Grades

This is the Form header title.

Upon clicking this button a user would be returned to the main menu form.

This data grid would display all the classes that belong to the logged in teacher as soon as the form loads. From here a teacher will be able to view all of the grades that the students within one of their classes achieved. They will also be able to edit any one of their student's grades and will be able to view all of a student's achieved grades to see how they performed in other subjects.

▼ Head 1	▼ Head 2	▼ Head 3	
Cell 1	Cell 2	Cell 3	
Cell 4	Cell 5	Cell 6	
Cell 7	Cell 8	Cell 9	
Cell 10	Cell 11	Cell 12	

Edit/Add Student

The screenshot shows a Windows-style application window titled "Edit Student Details". The window has standard minimize, maximize, and close buttons at the top right. Inside, there's a header bar with a "Return to Main Menu" button. Below this are ten input fields for student information: Firstname, surname, gender (with a dropdown arrow), ethnicity, attendance, date of birth, SNS status, free school meals (with a dropdown arrow), form, school year, and exams year. At the bottom are two buttons: "Edit/Add" and "Delete". Blue arrows from the right side point to various parts of the interface with explanatory text boxes.

This is the Form header title.

Upon clicking this button a user would be returned to the main menu form.

This selection of comb boxes and text fields will be automatically filled with the details of the record the user has selected to edit. However if a user selects to add a new record these fields will be blank. A strict set of validation will be used on these comb boxes and text fields to ensure data is consistent.

When this delete button is clicked the record that matches the current entered details will be deleted from the database.

When this edit button is clicked the record that was selected will be edited with the newly entered details in the database.

Add/Edit Teacher

This diagram illustrates the layout and functionality of an 'Edit TeacherDetails' form. The form header title is 'Edit TeacherDetails'. It features three buttons in the top right corner: a standard window control button, a square button, and a red 'X' button. Below the title, the form displays two text fields: 'Fistname' and 'surname'. At the bottom of the form are two buttons: 'Edit/Add' and 'Delete'. A large callout box on the left provides detailed information about the text fields. Arrows point from each annotation to its corresponding element on the form.

This is the Form header title.

Upon clicking this button a user would be returned to the main menu form.

This selection text fields will be automatically filled with the details of the record the user has selected to edit. However if a user selects to add a new record these fields will be blank. A strict set of validation will be used on these text fields to ensure data is consistent.

When this edit button is clicked the record that was selected will be edited with the newly entered details in the database.

When this delete button is clicked the record that matches the current entered details will be deleted from the database.

Edit Class

This diagram illustrates the 'Edit Class' form interface. At the top left is the header 'Edit Class'. To its right is a button labeled 'Return to Main Menu'. Below these are five input fields: 'Firstname', 'surname', 'Class Group', 'Subject', and 'Year'. Underneath these fields are two buttons: 'Edit/Add' and 'Delete'. A horizontal bar at the top right contains three icons: a minus sign, a square with a diagonal line, and an 'X'. A large blue arrow points from the 'Edit/Add' button to a callout box explaining its function. Another blue arrow points from the 'Delete' button to a callout box explaining its function. Callout boxes also point to the 'Edit Class' header, the 'Return to Main Menu' button, and the top bar icons.

This is the Form header title.

Upon clicking this button a user would be returned to the main menu form.

This selection of comb boxes and text fields will be automatically filled with the details of the record the user has selected to edit. However if a user selects to add a new record these fields will be blank. A strict set of validation will be used on these comb boxes and text fields to ensure data is consistent.

When this edit button is clicked the record that was selected will be edited with the newly entered details in the database.

When this delete button is clicked the record that matches the current entered details will be deleted from the database.

Edit Grade

The screenshot shows a Windows-style application window titled "Edit Grade". The window has a standard title bar with minimize, maximize, and close buttons. Inside, there's a toolbar with a "Return to Main Menu" button. The main area contains seven data entry fields: "Firstname" (text box), "surname" (text box), "Subject" (combobox), "Level" (combobox), "Grade" (combobox), "Grade Type" (combobox), and "Year" (combobox). At the bottom are two buttons: "Edit/Add" and "Delete". Blue arrows point from various parts of the interface to callout boxes with descriptions:

- An arrow points from the title bar to a callout box: "This is the Form header title."
- An arrow points from the "Return to Main Menu" button to a callout box: "Upon clicking this button a user would be returned to the main menu form."
- An arrow points from the "Edit Grade" window title to a callout box: "This selection of comb boxes and text fields will be automatically filled with the details of the record the user has selected to edit. However if a user selects to add a new record these fields will be blank. A strict set of validation will be used on these comb boxes and text fields to ensure data is consistent."
- An arrow points from the "Delete" button to a callout box: "When this delete button is clicked the record that matches the current entered details will be deleted from the database."
- An arrow points from the "Edit/Add" button to a callout box: "When this edit button is clicked the record that was selected will be edited with the newly entered details in the database."

Add Grade

This is the Form header title.

Upon clicking this button a user would be returned to the main menu form.

From this group box a user will be able to search students using a number of different filters. Then they will be able to select an individual student and view all of the selected students achieved grades in the data grid.

This group box will allow a user add a new grade to the system for the selected student. To do this they will have to select a valid set of data and then press the add grade button with all fields being completed.

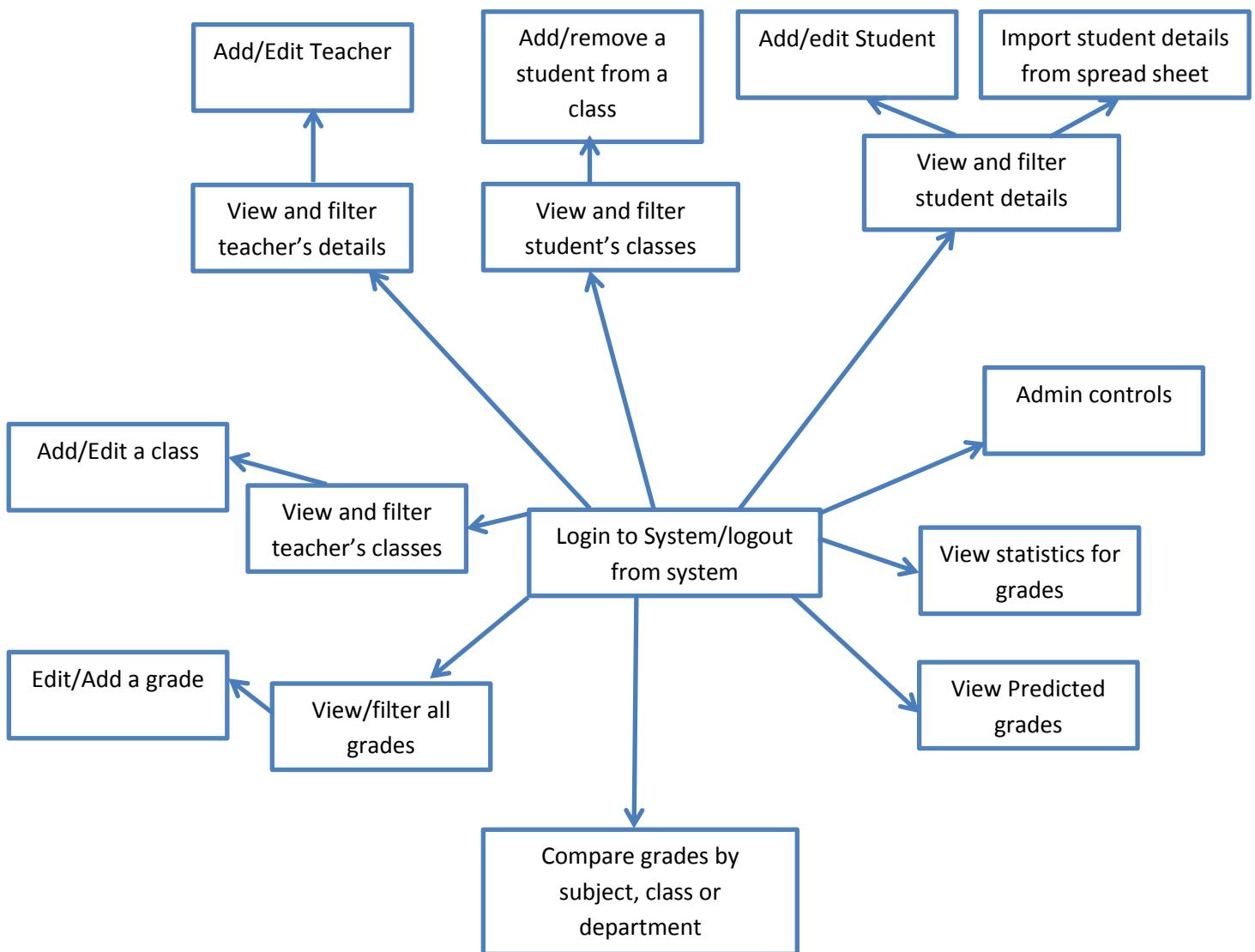
This data grid would display all the selected student's grade records that correspond to the user's academic year choice and match the search criteria.

Head 1	Head 2	Head 3	
Cell 1	Cell 2	Cell 3	
Cell 4	Cell 5	Cell 6	
Cell 7	Cell 8	Cell 9	
Cell 10	Cell 11	Cell 12	

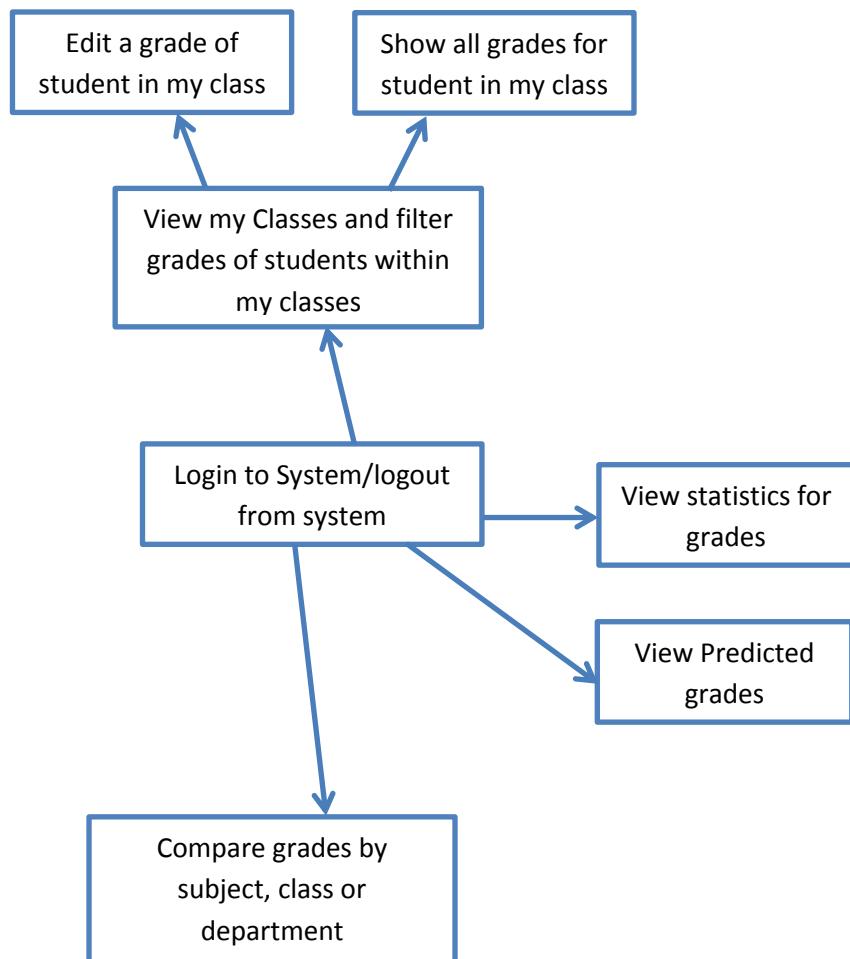
Form Navigation Design (admin-user)

Use of Breadcrumb trail

Within my initial HCI designs, I had chosen to use buttons to navigate between forms, however I have now decided that I will not use this as a method of navigation. Instead I will be implementing a “breadcrumb trail” style of navigation. This will show the navigation route a user has followed, along with the current page form the user is on. The user will then be able to navigate back to any form they have navigated through to reach their destination by clicking the correctly labelled hyperlink. I believe this is a much more professional and user friendly alternative to using buttons because it creates less clutter on a form and it is a design that would be familiar to a user because this style of navigation is currently used in many systems.



Form Navigation Design (non admin-user)



Security and Integrity of Data

Since there is lots of personal data that will be stored about students and staff within the system, it is very important that data is secure. Students will not have access to the system since the system will only ever be installed on staff accounts and staff machines, so having a restricted access profile between student and staff is not a concern. However some data should only be visible by administrative users such as student's personal details or teacher's personal details because only an administrator would ever require this information. Therefore, my system will have a user login form that determines whether the user that has logged in (assuming the login is valid) is an admin user or a non admin-user. If the user is an admin they will have access to an advanced set of features and will have unrestricted control of the system, whereas a non-admin user will only have certain privileges.

In order to protect the integrity of stored data within my system, all data entry forms and data entry fields will have strict validation rules; this will prevent incorrect data from being stored in the system. I will also validate user typographic errors. Any errors will be caught by the system to prevent it from crashing, and an appropriate error message will display to the user.

My data will all be stored within a local school networked server using Microsoft SQL Server Management Studio; I do not need to worry about my data being insecure here because the server itself is in a secure IT server room within the school. Furthermore to gain access to my database on the server a login username and password is required; this information will be in the software code which will be inaccessible after the program has been compiled. However, once my system is completed I will need to pass the details of my login and username access of the database to the primary users of my system. I also do not need to worry about data becoming corrupt on the server because in this case a backup copy of the databases data and relations would be restored since the school does weekly backups of all their server data.

Identification of Storage Media

I expect the final software executable file to be relatively small since it is being used for specific bespoke purposes; I estimate that the final executable file should be no more than 5 megabytes in space. It could be more than this though depending on the size of graphics used for the designing of forms. I am not sure about the size of how large my database will be but I am sure I do not need to worry about this since the SQL server is capable of storing 80 GB. Since my actual software executable will be very small I could consider many different options when distributing my system.

I could choose a CD-ROM or CD-R since data will only need to be read from discs therefore a disc format such as DVD-RW or CD-RW would be unnecessary. A CD-ROM is capable of storing between 600-700 megabytes of data. One advantage of a CD-ROM to distribute the system is that it can be re-used and installed on several systems one after another. However this could be considered a disadvantage because only one device can install the software at any one time. Furthermore, many of the teaching staff's laptops do not contain disk-drives because CD and DVD storage is slowly becoming an obsolete technology.

The software could also be distributed via USB flash-drive. There are several advantages to USB flash-drives. They are lightweight removable and rewriteable. They are widely used because they are compact, fast, can hold lots of data and are more reliable than other storage mediums due to their lack of moving

parts and robust design. The installation speed from USB 2.0 and USB 3.0 are also much faster than installation from disc. One drawback is that their small size means that they can be easily misplaced or lost. USB ports have become an industry standard for all new laptops and desktop machines and every teacher has access to a laptop or machine with USB ports, so this is one ideal reason to consider USB since it is widely compatible. They also do not require a networked connection for install but any user that wants to use my system will need an internet connection anyway because of the connection to the SQL server. One other major drawback of this choice of storage medium is that the software will have to be installed one machine at a time or lots of USB flash drives will have to be distributed (one for each teacher) which could be quite expensive.

My choice for the storage of the executable file would be from a secure location on the schools internal teacher portal which has a strict firewall which will protect unwanted users from accessing the executable along with authorisation and authentication requirements. The only problem with this choice is that a user would have to be connected to the internet to access the executable file, but this should not be a problem because in order to use the system a networked connection between the program and the SQL server must be established.

Sample of Possible SQL Queries

The below SQL queries will not represent queries that I will actually use within my system but are examples of different types of SQL queries I plan to use and expect I will have to use within my system to manipulate the data stored within the SQL server database for practical use by the system.

Displaying All Subjects in the Database

SELECT * FROM Subject

Description

The above query would be useful to select every single field from a single table if a user has not used any filtering.

Displaying All A grades from the grade relation

Select * FROM Grade Where Grade = 'A'

Description

The above query would be used to return all the fields from a table where records within that table match the criteria, in the above case this would be all records with a grade A.

Displaying only B grades from the grade relation

Select Grade FROM Grade Where Grade = 'B'

Description

The above query would be used to return selected fields from a table where records within that table match the criteria, in the above case this would be all records with a grade B.

Adding a new Subject into the database

INSERT INTO Subject (SubjectID, DepartmentID, Subject) VALUES (?,?,?)

Description

The above type of SQL query would be used in any instance where a user wants to create a new record within the SQL server database. It works by choosing a table to insert the new record and then inserting values under the corresponding table's fields.

Update an existing teacher's details

UPDATE Teacher Set TeacherID = '36' WHERE TeacherID = '29'

Description

The above type of SQL query would be used in any situation where an existing record within the system needs to be updated. In order to do this a table must be chosen and then the new values need to be set where there is a record matching the chosen criteria.

Delete a student from the system

DELETE FROM Student WHERE StudentID = '249'

Description

This above SQL query would be used in any scenario where a user wants to delete a record. In order to do this a table must be chosen and then an identifier must be used which matches a record.

Generate a pivot table of grades using a where identifier

SELECT [A], [B] FROM Grade PIVOT (COUNT(GradeID) FOR Grade IN ([A],[B])) AS PVTTABLE WHERE SubjectID = '1'

Description

This SQL query uses a pivot table to count the number of grades that match either A, B for a chosen subject. These numbers will then be written in a dynamic query table.

Sample of Planned Algorithms and Pseudocodes

Within this section I have planned some essential algorithms which I expect I will have to use within my system. I have included a name for each planned algorithm and a description of what the algorithm should be able to do when it is implemented within my solution properly. Furthermore, I have written pseudo code for my planned algorithms as I have tried to show how I will go about implementing my algorithms line by line. My pseudo code is loosely based on the programming languages Visual basic and SQL where possible since these will be my chosen programming structures used for my system.

Algorithm: Logging into System
Description
This algorithm will be used to determine whether a valid user has or has not attempted to log into the system. It will first check whether the entered details are valid and then it will check whether the logged in user is an admin or not. The user will then be given a set of privileges based on their admin status. If the username or password entered is not valid then the fields should be reset to blank.
Pseudo-code
<pre> Dim sqlquery Dim username Dim password Dim datatable Dim admin Sqlquery.connection=con Sqlquery = "SELECT LoginUsername, Password FROM UserLogin WHERE LoginUsername = '" & usernamefield.Text & "' AND Password = '" & passwordfield.Text & "'" Datatable.fill USING sqlquery If rows = true THEN Password=check.datatable.(“password”) username=check.datatable.(“username”) admin=check.datatable.(“admin”) DISPLAY mainmenu ELSE Output ("Invalid Username or Password") Usernamefield.text = "" Passwordfield.text = "" End if </pre>

Algorithm: Connecting to the database
Description
This algorithm connects the system to the SQL Server database through a connection string that is set by the program. This connection string allows a SQL connection to be made that uses the login credentials of the connection string to access the database. The algorithm first attempts to establish a connection using the connection string which has been set, if a connection cannot be established then an appropriate message is displayed to the user.
Pseudo-code
Procedure DatabaseConnect() Imports System.SQL Procedure Databaseconnect () Dim Connectionstring Dim con Dim sql Dim connection Connection string = "Database Connection Path here" Con = SQLconnection USING connectionstring Try Con.open Catch exception Output("Can not open SQL Server connection! Make Sure Server Device is Switched on. ") End try End Procedure

Algorithm: Objects visible states
Description
This algorithm is used to display different objects on the form depending on the Case of a field as it is changed. When the Case value Matches the field value then the corresponding Case runs its code. For example if the field.text value is "Name" then all code within the name case will run.
Pseudo-code
Procedure handles field.indexchanged Select Case "Selected-Field" Case "" Namefield = non-visible Formfield = non-visible Yearfield= non-visible displayStudentgroup = non-visible Valueuelabel = non-visible Case "Name" Namefield = Visible Formfield = non-visible Yearfield= non-visible displayStudentgroup = Visible Valueuelabel = Visible Valueuelabel.text = "Name :" Case "Form" Generatecombobox(Searchform) Using ("Student", "Form") Namefield = non-visible Formfield = Visible Yearfield= non-visible displayStudentgroup = Visible Valueuelabel = Visible Valueuelabel.text = "Form :" End Select End Procedure

Algorithm: Generate a Combobox item selection
Description
This algorithm is set as a function so that a dynamic table, field and needemptyvariables can be passed into the function. Once a table and field are passed into the function a SQL statement is used to return records using the table and field values. These records are then stored within a dataset table. The dataset table is then set as the filterdt property, which is the data table returned once the function has been processed. If the Boolean state of needempty is true then a new row is inserted at the row position 0 to the filterdt data table.
Pseudo-code
Procedure GenerateCombo(table, field, needempty) Sql = "SELECT DISTINCT " & field & " FROM " & table & " ORDER BY " & field & " ASC" Dim FilterDS Dim FilterDT Datadaptor = New SQLdataadaptor USING (sql,con) Datadaptor.fill(FilterDS, table) Filterdt= Filterds If needempty = true then Insert New Row at filterDt.rows(0) End If Return FilterDT End Procedure

Algorithm: GCSE Grade Weighting algorithm
Description
This algorithm Sets the grade weighting of a GCSE subject and returns the value of the grade weighting so that it can be applied to the corresponding grade. These weightings are particularly important because they are used to calculate a varied range of realistic grade predictions based on the relative difficulties of the subjects the student has taken at GCSE level.
Pseudo-code
Procedure GCSEWeighting(l,dt) Dim GCSEweighting If dt.rows(i)(“subject”) = “Chemistry” Then GCSEweighting = 1.22 End if If dt.rows(i)(“subject”) = “Physics” Then GCSEweighting = 1.19 End if If dt.rows(i)(“subject”) = “Biology” Then GCSEweighting = 1.18 End if ... Return GCSEweighting End Procedure

Algorithm: Generating the A2 predicted grade	Location: Module1
Description	
<p>This function is used to generate a unique A2 predicted grade for a subject depending on a student's average point score and the subject the grade is predicted for. A weighting value is applied to the average point score of a student to determine the predicted grade of an A2 subject depending on its relative difficulty. Once the point score has been calculated, it is then processed by a second algorithm within this function which determines the actual A2 predicted grade by converting the subject score to a grade. This varies depending on the point score so that a range of grades are achieved depending on the average point score of the student.</p>	
Pseudo-code <pre> Procedure A2Predictedgrade(subject, l, dt, averagepointscore) Dim predictedgrade Dim subjectpointscore Subjectpointscore=averagepointscore If dt.rows(i)(“subject”) = “Chemistry” Then Subjectpointscore = averagepointscore * 0.83 End if If dt.rows(i)(“subject”) = “Physics” Then Subjectpointscore = averagepointscore * 0.85 End if If dt.rows(i)(“subject”) = “Biology” Then Subjectpointscore = averagepointscore * 0.85 End if ... If subjectpointscore >= 58 then Predictedgrade = “A*” Elseif subjectpointscore < 58 and subjectpointscore >= 52 then Predictedgrade = “A” Elseif subjectpointscore < 52 and subjectpointscore >= 46 then Predictedgrade = “B” Elseif subjectpointscore < 46 and subjectpointscore >= 40 then Predictedgrade = “C” Elseif subjectpointscore < 40 and subjectpointscore >= 34 then Predictedgrade = “D” Elseif subjectpointscore < 34 and subjectpointscore >= 28 then Predictedgrade = “E” Elseif subjectpointscore < 28 then Predictedgrade = “U” End if Return Predictedgrade End Procedure </pre>	

Algorithm: Validating entry fields	Location: Edit Classes
Description	
This function validates whether a set of textbox or combo box entry fields are not empty and that they have values selected by the user. If all fields have been completed then the function returns a true state. However if any field has not been completed then a false state is returned. Also fields that have not been completed have their background colours changed to a red colour to indicate this to the user. After a field has been completed its background is returned to a white colour.	
Pseudo-code	
<pre>Procedure txtvalidate() Txtvalidate =true If firstnamefield. Text = "" Then Firstnamefield.background = red Txtvalidate =false Else Firstnamefield.background = white End if If surnamefield. Text = "" Then surnamefield.background = red Txtvalidate =false Else surnamefield.background = white End if If classfield. Text = "" Then classfield.background = red Txtvalidate =false Else classfield.background = white End if ... End Procedure</pre>	

Algorithm: Exporting data grid data
Description
This algorithm is responsible for building a CSV formatted string using the contents of a data grid that is being displayed on a form. Once the CSV style string has been written, a user is then able to select where to save a file and is also given the option of naming the file. From here the CSV formatted string is written to the new file the user has named in the desired location the user has set.
Pseudo-code
Procedure ExportCSV() Dim csv For each datagridviewcolumn in "datagrid".columns Csv += column(headertext) & "," Next Csv.new line For each datagridviewrow in "datagrid".rows For each datagridviewcell in "datagrid".rows.cells Csv += cell.text & "," ";" & "," Next Csv.new line Next Dim savefiledialog As save filedialog() svaefiledialog.file.filterselection = "Spreadsheet CSV file (*.csv) *.csv" savefieldialog.restoredirectory = true if savefiledialog.result = ok then Dim SW as streamwriter using savefiledialog.openfile If SW is not Nothing then Sw.write(csv) Sw.close() End if End if

Algorithm: Importing CSV file of records
Description
With this algorithm a user will be able to select an academic year and then import a CSV formatted spread sheet file into the database system.
Pseudo-code
Dim openfiledialog Dim dataset Dim filepath Dim datagrid Dim datatable Dim sqlquery Dim rows Dim x If openfiledialog.selection = ok Then Filepath = GET FILE PATH Dim connectionstring USING filepath AND USING Delimiter OR Splitstring Connectionstring.open Sqlquery = "SELECT * FROM " & filepath & " Datagrid.fill USING sqlquery Datagrid.addnewcolumn = ("inserted (true/false)") Rows = datagrid.rowcount X=0 For datagrid.rowcount = 0 to datagrid.rowcount = rows Sqlquery = " INSERT INTO Student Values (datagrid.row(x)(0), datagrid.row(x)(1) ... datagrid.row(x)(9), X = x + 1 Next Output ("Insert Data successful") Else Output ("please select an academic year to import student details for") End if

Testing Plan

Testing Strategies to consider

Bottom-up testing

This type of testing is most likely not going to be a good test strategy for myself since I will be adding features to my system as I go along and I will not purely be using a bottoms-up implementation approach.

Top-down testing

This type of testing could be of more use to me since I will mainly be implementing my system in a top-down approach starting with one or two forms and then working downwards from there developing my system as I proceed with my implementation.

Dry Running

This test strategy involves tracing code that is read through and checked by hand. A trace table is completed to check the values of variables at every stage of execution. One disadvantage of dry running is that it is very time consuming and since I have strict time constraints it may not be something I can consider. Dry running is capable of revealing many possible causes for error, these can include:

- Variables not being declared and initialised correctly
- Loop statements not terminating
- Boolean expressions not being correct
- Parameters not being supplied in the correct order

Black Box testing

Black box testing is a method of software testing that examines the functionality of an application without peering into its internal structures. In this type of testing, individual components of various forms will be tested for errors. This is the most appropriate form of testing for my system because it should be able to both small and large bugs within my system that could not be detected without testing components individually.

Choice of test data types

- **Normal Data (T):** I will use typical valid data examples to test the valid functionality of my system.
- **Erroneous Data (E):** I will use non-valid data examples to test data validation and verification within my system so that I can see how the system responds and if it does so appropriately.
- **Exceptional values (X):** where possible I will try to test my program using exceptional data such as leaving text fields blank and using non valid data selection. I will also use boundary data, a test data selection technique in which values are chosen to lie along data extremes. I will do this so that I can see how my system responds to extreme errors.

Input and Output Testing Design

In order to ensure that my system will be able to correctly handle all types of user input (**TEX**), I have planned several testing procedures which will test whether algorithms will execute correctly. I have also planned a format in which I will use to test the navigation links of my form.

The following table below is designed to compare what I believe will be the expected outcome of my tests against the actual outcome of my tests. Within the input data/action column I will include the TEX (typical, erroneous, and exceptional) data indicators to show what types of data I am testing with a test. I will use the Actual outcome column to reference to screenshot evidences of tests. Below is a sample of functions I expect that I will have to test within my program, however once my program has been developed I'm sure that there will be lots more functions and components that will actually require testing.

Test No.	Input Data/Action	Expected Outcome	Actual Outcome	Comments and Actions
1	T: Login with valid login credentials.	I expect the User to be directed to the Main Menu form.	N/A	N/A
	E: login with incorrect login credentials.	I expect the user to remain on the login form and the values of the text fields to be reset to blank. Also an error message should be displayed.	N/A	N/A
	X: login with blank fields.	The user should remain on the Login form and the incorrect user message should display.	N/A	N/A
2	Test filtering a student's selection.	I expect the student's that have been filtered to be returned in a data grid.	N/A	N/A
3	Test when a user selects to edit a student's details.	I expect the user to be able to edit the details of the selected student from a new form.	N/A	N/A
4	Test when a user selects to add a new student to the database.	I expect the user to be able to add a new student to the database system from a new form.	N/A	N/A
5	Editing a student's details.	I expect the user to be able to edit a student's details and for the changes to be made to the database.	N/A	N/A
6	Adding a student's details.	I expect the user to be able to add a new student's details and for the changes to be made to the database.	N/A	N/A
7	Test filtering a teacher's details.	I expect the teachers that have been filtered to be	N/A	N/A

		returned in a data grid.		
8	Test when a user selects to edit a teacher's details.	I expect the user to be able to edit the details of the selected teacher from a new form.	N/A	N/A
9	Test when a user selects to add a new teacher to the database.	I expect the user to be able to add a new teacher to the database system from a new form.	N/A	N/A
10	Editing teacher's details.	I expect the user to be able to edit a teacher's details and for the changes to be made to the database.	N/A	N/A
11	Adding teacher's details.	I expect the user to be able to add a new teacher's details and for the changes to be made to the database.	N/A	N/A
12	Test the functionality of the select academic year function on the main form.	I expect the current academic year selection to affect the system's filtering selections.	N/A	N/A
13	Test filtering a student's classes.	I expect the student's classes that have been filtered to be returned in a data grid.	N/A	N/A
14	Test when a user selects to add a student to a class.	I expect the user to be able to add a student to the selected class from a new form.	N/A	N/A
15	Test when a user selects to remove a student from a class.	I expect the user to be able to remove the student's class record from the database system on a new form.	N/A	N/A
16	Removing student from class.	I expect the user to be able to remove the student from the selected class and for the changes to be made to the database.	N/A	N/A
17	Adding student to a class.	I expect the user to be able to add a student to the selected class and for the changes to be made to the database.	N/A	N/A
18	Test filtering a teacher's classes.	I expect the teacher's classes that have been filtered to be returned in a data grid.	N/A	N/A
19	Test when a user selects to edit a class.	I expect another form to open allowing the user to edit details of the selected	N/A	N/A

		class.		
20	Editing a class' details.	I expect the user to be able to edit the selected class and for the changes to be made to the database.	N/A	N/A
21	Test selecting predicted grades for a student, class, form or year group.	I expect a list of predicted A2 grades to be returned for the selected student's selection.	N/A	N/A
22	Test filtering a selection of grades stored within the system database.	I expect the grades within the database to be filtered by the user's selection and returned in a data grid.	N/A	N/A
23	Test adding a grade for a selected student.	I expect a user to be able to add a grade for a selected student.	N/A	N/A
24	Test when a user selects to edit a grade.	I expect a form to open which allows the user to edit the selected grade.	N/A	N/A
25	Editing a grade.	When a selected grade is edited, the same changes must be made to the database.	N/A	N/A
26	Test filtering to view grade percentages of achieved and predicted grades by class, year, department or subject.	I expect a graph to display which represents the user's selection alongside the raw data for the user's selection.	N/A	N/A

I will use the following test evidence reference table to provide evidence for my tests and explain what the evidence shows/represents in the description column. The reference number for a test will be provided in the actual outcome column of the testing table.

Ref No.	Description	Picture Evidence

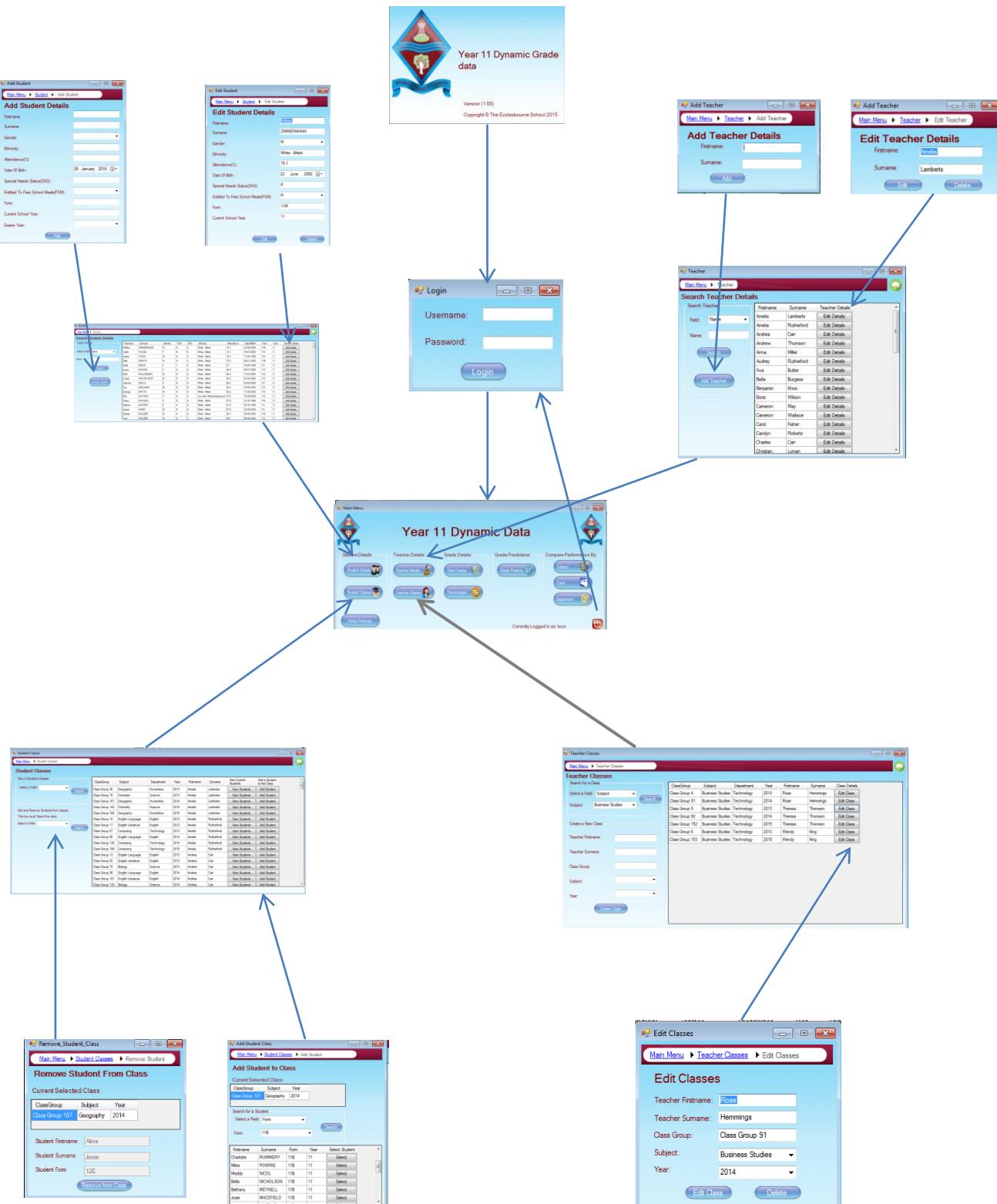
Navigation Testing Design

In order to test the navigation of my system and to be able to see which forms are linked to each other I've created a testing table. The testing table is unidirectional, which means that one form can link to another (e.g. A to B) but this does not always mean that the user can navigate back to the original form (e.g. B to A). To display a successful navigation test in the table designed below, a series of ticks or crosses will be used. As I am not too sure how many forms I will have by end development the table below is just an example and will most likely not represent the size of the table I come to use later. I will use the key below:

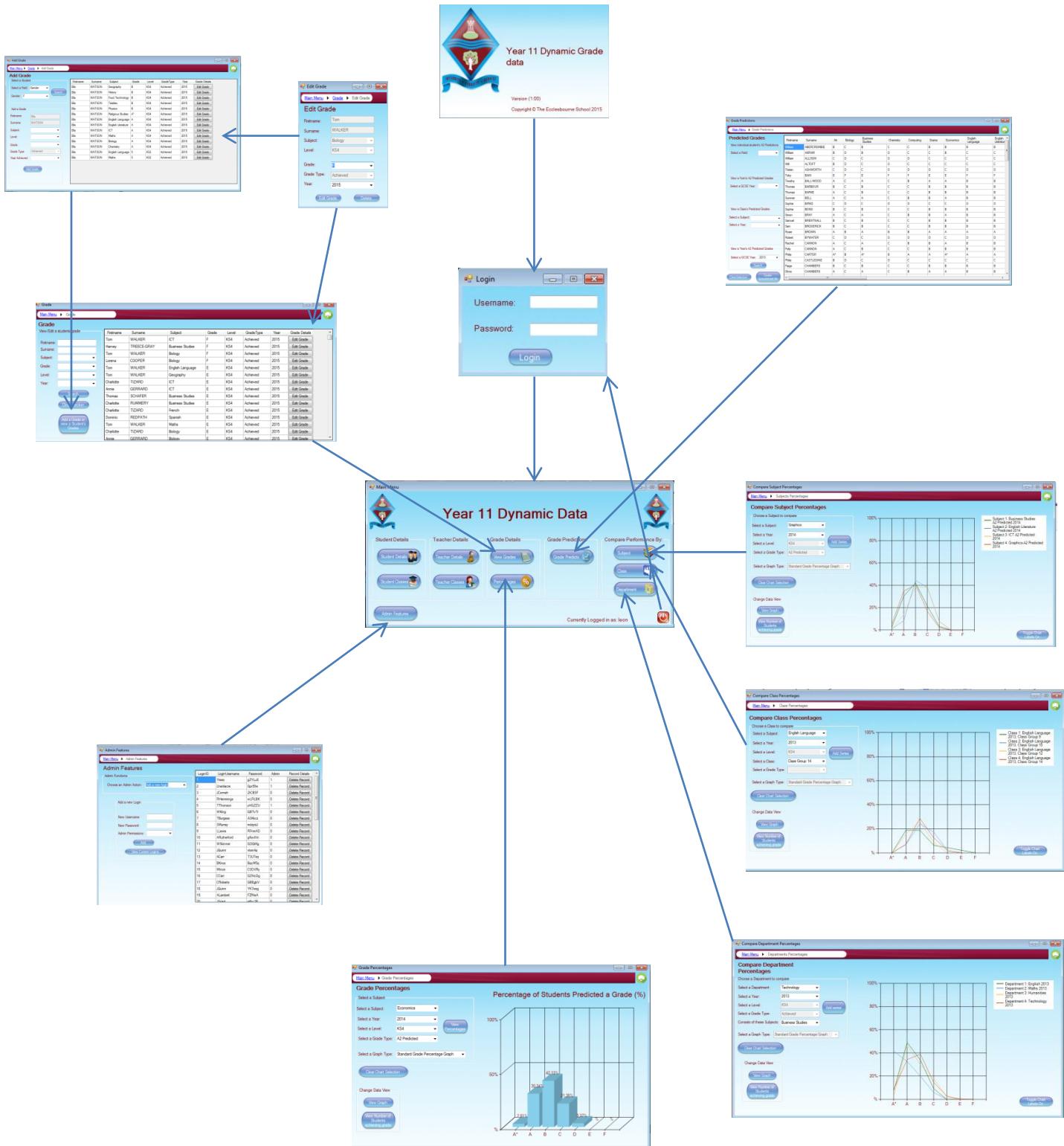
Successful Navigation	
Unsuccessful Navigation	
No need for Navigation	
Forms are not Linked	

Navigating From: →	A	B	C	D	E	F
Navigating To: ↓						
A						
B						
C						
D						
E						
F						

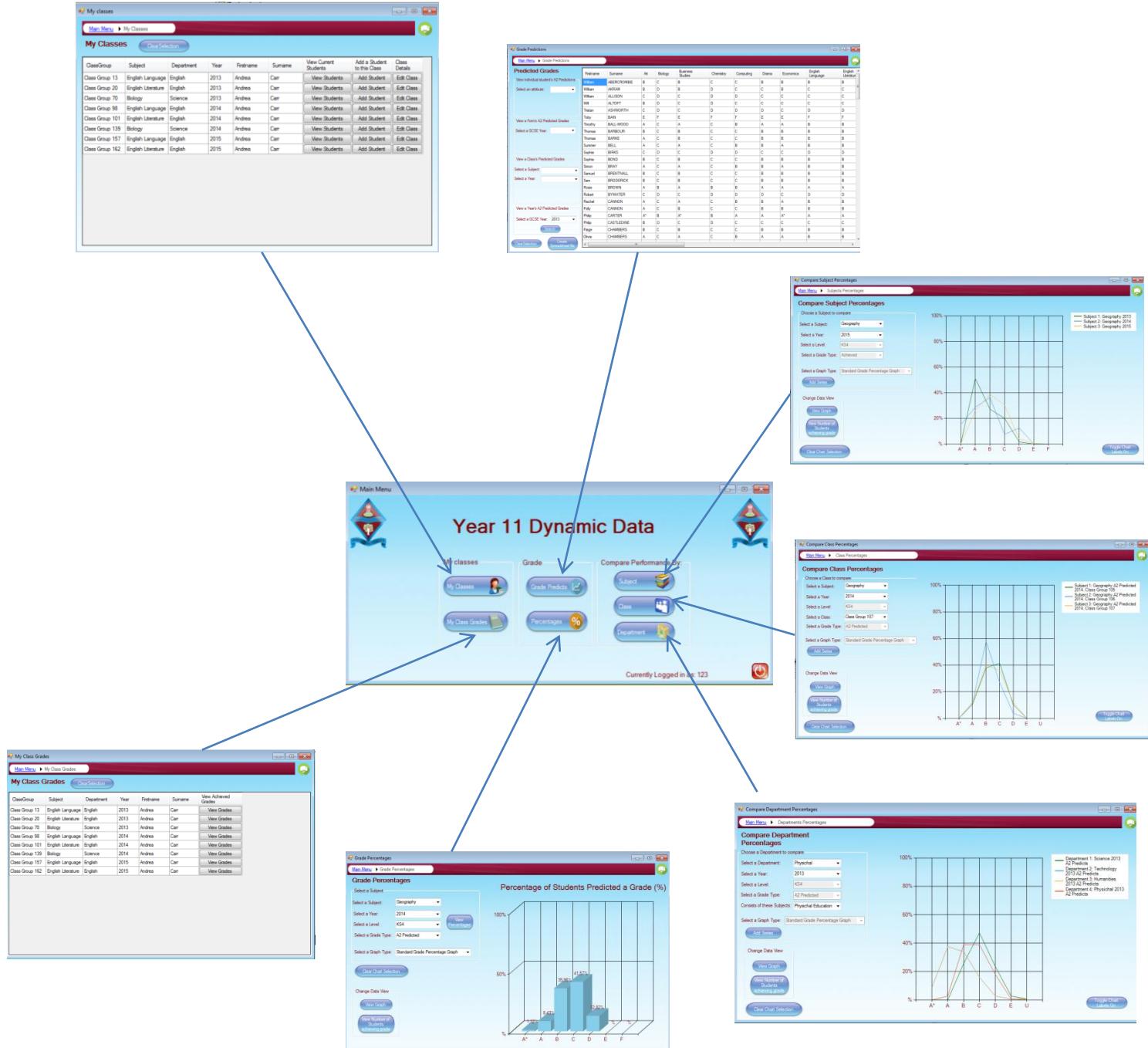
Form Navigation Overview (admin user)



Form Navigation Overview – continued (admin user)



Form Navigation Overview – continued (non-admin user)



Database Creation

In order to asset up my database within Microsoft SQL Server Management Studio, I first needed to manipulate the data that was provided by my end user. The data I was provided with consisted of student personal data and student grade data. For security and data protection reasons the data was hidden by mixing the names of students up. However, I also needed to have data within my database such as subjects, grade types and level types in regards to grades data. When implementing my tables the data I was provided with was in an excel document so I first of all had to create all my tables within Microsoft Excel before then inserting all of that data into the SQL Server. When creating my tables I referred closely to the normalised relations I had designed with my design section to make sure that all final data stored within the system would be atomic.

The screenshot below is an example of the Student relation with three year sets of student's data, provided by my primary users of the system. I had to rename the field headings to the ones I had designed earlier. Following this screenshot are all of the other table relations of data I had to create within my Microsoft Excel document:

Student relation

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	Student Firstname Surname	Gender	Entitled to free school	Special Needs	Ethnicity	Attendance	Date of Birth	Date of Birth							Date Of Birth	Form	Year	dataset	
2	William ZIMMERMAN M	K	Y	No	White - British	75.1	23/06/2000 00:00:00	23/06/2000 00:00:00	24/06/2000	23	06	2000	2000-06-23	1M	11	1	INSI		
3	Zoe JONES F	N	Y	No	White - British	74.7	18/07/1999 00:00:00	18/07/1999 00:00:00	19/07/1999	16	07	2000	2000-07-16	1H	11	1	INSI		
4	James YATES M	N	Y	No	White - British	73.0	19/09/1999 00:00:00	19/09/1999 00:00:00	19/09/1999	11	08	1999	1999-09-11	1H	11	1	INSI		
5	Matt WORTH M	K	Y	No	White - British	73.0	09/06/2000 00:00:00	09/06/2000 00:00:00	09/06/2000	09	01	2000	2000-01-09	1M	11	1	INSI		
6	Kirsty WOOD F	Y		No	White - British	81	15/09/1998 00:00:00	15/09/1998 00:00:00	15/09/1998	15	03	1999	1999-09-15	1UJ	11	1	INSI		
7	Tom COOM F	N	Y	No	White - British	82.5	28/06/2000 00:00:00	28/06/2000 00:00:00	29/06/2000	29	01	2000	2000-01-29	1S	11	1	INSI		
8	T Mall WILLIAMSON M	N	Y	No	White - British	88.2	17/03/2000 00:00:00	17/03/2000 00:00:00	17/03/2000	17	03	2000	2000-03-17	1D	11	1	INSI		
9	Jordan WHITEMISTER M	N	Y	No	White - British	86.2	05/04/2000 00:00:00	05/04/2000 00:00:00	05/04/2000	03	05	2000	2000-05-03	1S	11	1	INSI		
10	Alannah VELLS F	N	Y	No	White - British	86.2	05/04/2000 00:00:00	05/04/2000 00:00:00	05/04/2000	05	04	2000	2000-04-05	1F	11	1	INSI		
11	Wesley WILKINSON M	N	Y	No	White - British	85.2	05/05/2000 00:00:00	05/05/2000 00:00:00	05/05/2000	05	05	2000	2000-05-05	1D	11	1	INSI		
12	George VATTS M	N	Y	No	White - British	88.2	19/06/2000 00:00:00	19/06/2000 00:00:00	19/06/2000	11	06	2000	2000-06-11	1D	11	1	INSI		
13	Ella VATSON F	K	Y	Any other White background	87.5	19/06/2000 00:00:00	19/06/2000 00:00:00	19/06/2000	18	08	2000	2000-08-19	1H	11	1	INSI			
14	Eliza VATSON F	N	Y	Any other White background	87.3	12/09/1999 00:00:00	12/09/1999 00:00:00	12/09/1999	12	10	1999	1999-10-12	1H	11	1	INSI			
15	Poppy VATSON F	N	Y	Any other White background	87.5	20/09/1999 00:00:00	20/09/1999 00:00:00	20/09/1999	20	09	1999	1999-09-20	1U	11	1	INSI			
16	James VARD M	N	Y	No	White - British	87.9	18/05/2000 00:00:00	18/05/2000 00:00:00	18/05/2000	18	05	2000	2000-05-18	1L	11	1	INSI		
17	Nathan WALKER M	N	Y	No	White - British	89.7	30/06/2000 00:00:00	30/06/2000 00:00:00	30/06/2000	30	08	2000	2000-08-30	1B	11	1	INSI		
18	Tom WALKER M	S	Y	No	White - British	88.7	08/07/2000 00:00:00	08/07/2000 00:00:00	08/07/2000	08	02	2000	2000-07-08	1S	11	1	INSI		
19	Hayley WALLER F	N	Y	No	White - British	83.7	12/03/2000 00:00:00	12/03/2000 00:00:00	12/03/2000	12	01	2000	2000-01-12	1F	11	1	INSI		
20	Amelia YAN F	N	Y	No	White - British	88.7	14/06/2000 00:00:00	14/06/2000 00:00:00	14/06/2000	14	02	2000	2000-06-14	1S	11	1	INSI		
21	Dulcie UPCHURCH F	N	Y	No	White - British	89.7	23/07/1999 00:00:00	23/07/1999 00:00:00	23/07/1999	28	09	1999	1999-09-28	1H	11	1	INSI		
22	Harvey UPCHURCH F	K	Y	No	White - British	89.7	16/09/1999 00:00:00	16/09/1999 00:00:00	16/09/1999	03	11	1999	1999-10-03	1S	11	1	INSI		
23	Charlotte UPCHURCH M	N	Y	No	White - British	89.7	07/10/2000 00:00:00	07/10/2000 00:00:00	07/10/2000	07	07	2000	2000-07-07	1H	11	1	INSI		
24	Loren TOMPKINSON F	N	Y	No	White - British	89.7	23/07/1999 00:00:00	23/07/1999 00:00:00	23/07/1999	28	09	1999	1999-09-28	1H	11	1	INSI		
25	Oliver TOMPKINSON M	N	Y	No	White - British	91.4	06/04/2000 00:00:00	06/04/2000 00:00:00	06/04/2000	06	04	2000	2000-04-06	1L	11	1	INSI		
26	Charlie TOWNSEND M	Y		No	White - British	91.4	06/05/2000 00:00:00	06/05/2000 00:00:00	06/05/2000	06	05	2000	2000-05-06	1H	11	1	INSI		
27	Charlotte TOWNSEND M	M	Y	No	White - British	91.4	29/03/2000 00:00:00	29/03/2000 00:00:00	29/03/2000	23	03	2000	2000-03-23	1H	11	1	INSI		
28	Imogen THORNBURN F	K	Y	No	White - British	91.4	03/07/1999 00:00:00	03/07/1999 00:00:00	03/07/1999	12	03	1999	1999-12-03	1L	11	1	INSI		
29	Colette THOMPSON F	Y		No	White - British	91.4	19/09/1999 00:00:00	19/09/1999 00:00:00	19/09/1999	19	09	1999	1999-09-19	1H	11	1	INSI		
30	Sam THOMPSON M	N	Y	No	White - British	91.4	06/04/2000 00:00:00	06/04/2000 00:00:00	06/04/2000	06	04	2000	2000-04-06	1L	11	1	INSI		
31	Diana THOMPSON F	N	Y	No	White - British	91.4	20/09/2000 00:00:00	20/09/2000 00:00:00	20/09/2000	20	01	2000	2000-01-20	1S	11	1	INSI		
32	Dan TAYLOR M	N	Y	Any other White background	91.4	19/07/1999 00:00:00	19/07/1999 00:00:00	19/07/1999	19	11	1999	1999-11-19	1H	11	1	INSI			
33	Joseph TAYLOR M	K	Y	No	White - British	93.1	29/12/1998 00:00:00	29/12/1998 00:00:00	29/12/1998	29	12	1999	1999-12-29	1L	11	1	INSI		
34	Harry TAYLOR M	N	Y	No	White - British	93.1	04/04/2000 00:00:00	04/04/2000 00:00:00	04/04/2000	04	04	2000	2000-04-04	1H	11	1	INSI		
35	Jasmine TAYLOR M	N	Y	No	White - British	93.1	16/01/2000 00:00:00	16/01/2000 00:00:00	16/01/2000	16	01	2000	2000-01-16	1S	11	1	INSI		
36	James SUMMERS M	N	Y	No	White - British	93.1	08/05/2000 00:00:00	08/05/2000 00:00:00	08/05/2000	08	02	2000	2000-05-08	1L	11	1	INSI		
37	Lauren SUMMERS M	N	Y	No	White - British	93.1	17/09/2000 00:00:00	17/09/2000 00:00:00	17/09/2000	17	05	2000	2000-05-17	1B	11	1	INSI		
38	Sammy SUMMERS F	N	Y	No	White - British	93.1	05/09/2000 00:00:00	05/09/2000 00:00:00	05/09/2000	05	11	1999	1999-12-05	1U	11	1	INSI		
39	Alex STEVART M	N	Y	No	White - British	93.1	05/05/2000 00:00:00	05/05/2000 00:00:00	05/05/2000	05	05	2000	2000-05-05	1S	11	1	INSI		
40	Charlotte STEVENS F	K	Y	No	White - British	93.1	24/04/2000 00:00:00	24/04/2000 00:00:00	24/04/2000	24	04	2000	2000-04-24	1S	11	1	INSI		
41	Chloe STEVENS F	N	Y	No	White - British	93.1	17/09/2000 00:00:00	17/09/2000 00:00:00	17/09/2000	17	05	2000	2000-05-17	1F	11	1	INSI		
42	Charlotte STEVENS F	M	Y	No	White - British	93.1	29/03/2000 00:00:00	29/03/2000 00:00:00	29/03/2000	23	03	2000	2000-03-23	1H	11	1	INSI		
43	Ffion SPENCER F	N	Y	No	White - British	93.1	26/01/2000 00:00:00	26/01/2000 00:00:00	26/01/2000	28	01	2000	2000-01-28	1H	11	1	INSI		
44	Ben SPENCER M	N	Y	No	White - British	93.1	09/04/2000 00:00:00	09/04/2000 00:00:00	09/04/2000	01	04	2000	2000-04-01	1L	11	1	INSI		
45	Martha SMITH F	N	Y	No	Pakistani	93.1	12/07/1999 00:00:00	12/07/1999 00:00:00	12/07/1999	12	11	1999	1999-11-12	1H	11	1	INSI		
46	Charlotte SPEDLEY F	N	Y	No	White - British	93.1	08/03/2000 00:00:00	08/03/2000 00:00:00	08/03/2000	10	02	2000	2000-03-10	1H	11	1	INSI		
47	Hariet SKINNER F	S	Y	No	White - British	93.1	13/08/2000 00:00:00	13/08/2000 00:00:00	13/08/2000	13	08	2000	2000-08-13	1L	11	1	INSI		
48	Dexter SIBTHORP-QI M	N	Y	No	White - British	93.1	10/09/2000 00:00:00	10/09/2000 00:00:00	10/09/2000	10	05	2000	2000-05-10	1UJ	11	1	INSI		
49	Calum SIBTHORP-QI M	K	Y	No	White - British	93.1	22/09/2000 00:00:00	22/09/2000 00:00:00	22/09/2000	22	08	2000	2000-09-22	1L	11	1	INSI		
50	Thomas SCHAFFNER M	K	Y	No	White - British	93.1	29/03/2000 00:00:00	29/03/2000 00:00:00	29/03/2000	23	01	2000	2000-03-23	1H	11	1	INSI		
51	Samuel RUSSELL M	N	Y	No	White - British	93.1	26/04/2000 00:00:00	26/04/2000 00:00:00	26/04/2000	26	04	2000	2000-04-26	1D	11	1	INSI		
52	Charlotte RUMMERY F	M	Y	No	White - British	93.1	13/06/2000 00:00:00	13/06/2000 00:00:00	13/06/2000	13	03	2000	2000-03-13	1B	11	1	INSI		
53	Bradley ROVETT M	K	Y	No	White and Black Caribbean	93.1	09/05/2000 00:00:00	09/05/2000 00:00:00	09/05/2000	08	03	2000	2000-05-08	1L	11	1	INSI		
54	Charlotte ROVETT M	N	Y	No	White - British	93.1	04/06/2000 00:00:00	04/06/2000 00:00:00	04/06/2000	05	05	2000	2000-05-05	1S	11	1	INSI		

Dataset Relation

Since my system would only store KS2 and KS4 grade data I just manually typed these into the table relation within my Microsoft SQL Server Database. I do store KS2 grade data within my system because my end user provided me with it but I do not however use it for grade predictions.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	Level ID	Level															
2		1 ks2															
3		2 ks4															
4																	
5																	
6																	
7																	
8																	
9																	
10																	
11																	
12																	
13																	
14																	
15																	
16																	
17																	
18																	
19																	
20																	
21																	
22																	
23																	
24																	
25																	
26																	
27																	
28																	

Subject Relation

Within the Subject relation I had to normalize the data my client had provided me and then put these subjects into a separate table relation that also link to the department table relation.

A	B	C
1	Subject ID	Department ID
2	1	8 Art
3	2	5 Business Studies
4	3	8 Drama
5	4	2 Economics
6	5	1 English Language
7	6	1 English Literature
8	7	5 Food Technology
9	8	2 Geography
10	9	2 Health And Social care
11	10	2 History
12	11	5 ICT
13	12	2 Leisure and Tourism
14	13	3 Maths
15	14	7 French
16	15	7 Spanish
17	16	7 German
18	17	5 Graphics
19	18	8 Music
20	19	2 Religious Studies
21	20	5 Resistant Materials
22	21	6 Physical Education
23	22	5 Computing
24	23	4 Biology
25	24	4 Chemistry
26	25	4 Physics
27	26	5 Textiles
28		

Department Relation

The department relation contains all the departments/faculties within the school, and these are used within the system to identify different subject areas.

A	B	C
1	Department ID	Department
2		1 English
3		2 Humanities
4		3 Maths
5		4 Science
6		5 Technology
7		6 Physical
8		7 Languages
9		8 Arts
10		
11		
12		
13		

Teacher Relation

Within the Teacher relation I had to create 50 random Teachers into a separate table relation that also link to the User Login table relation.

	A	B	C	D
1	Teacher ID	Login ID	First Name	Surname
2	1	1	Yvonne	Rees
3	2	2	Lily	Hardacre
4	3	3	Joanne	Cornish
5	4	4	Rose	Hemmings
6	5	5	Theresa	Thomson
7	6	6	Wendy	King
8	7	7	Trevor	Burgess
9	8	8	Stewart	Murray
10	9	9	Luke	Lewis
11	10	10	Amelia	Rutherford
12	11	11	William	Skinner
13	12	12	John	Quinn
14	13	13	Andrea	Carr
15	14	14	Benjamin	Knox
16	15	15	Madeleine	Ince
17	16	16	Charles	Carr
18	17	17	Carolyn	Roberts
19	18	18	Jonathan	Quinn
20	19	19	Amelia	Lambert
21	20	20	Julia	Grant
22	21	21	Sebastian	White
23	22	22	Andrew	Thomson
24	23	23	Audrey	Rutherford
25	24	24	Cameron	Wallace
26	25	25	Ryan	Powell
27	26	26	Ella	Stewart
28	27	27	Sam	Churchill
29	28	28	Jonathan	Morrison
30	29	29	Tracey	Hardacre
31	30	30	Christian	Lyman

User Login Relation

Within the User login relation I had to use an online random password generator to generate a list of password which I then assigned to each teacher.

	A	B	C	D
1	Login ID	Login Username	Password	Admin
2	1	Yrees	gZYLuE	1
3	2	Lhardacre	Gpr5Sw	1
4	3	JCornish	ZrCBsF	0
5	4	RHemmings	wLRLBK	0
6	5	TThomson	yHGZUU	1
7	6	WKing	QBTvTr	0
8	7	TBurgess	A3XKcz	0
9	8	SMurray	mdzpdJ	0
10	9	LLewis	RFnmAD	0
11	10	ARutherford	gKw4Vn	0
12	11	WSkinner	GDDQtkg	0
13	12	JQuinn	vtxmAp	0
14	13	ACarr	T3UTsq	0
15	14	BKnox	BqCMsq	0
16	15	MInce	CSDVRy	0
17	16	CCarr	GZHcDg	0
18	17	CRoberts	QBEBgbv	0
19	18	JQuinn	YK7wsg	0
20	19	ALambert	FZfNeA	0
21	20	JGrant	mfhc3B	0
22	21	SWhite	E3MW3C	0
23	22	ATHomson	MJrPsq	0
24	23	ARutherford	w7x6MC	0
25	24	CWallace	BzDC6f	0
26	25	RPowell	f86Bbu	0
27	26	ESTewart	p6EFqZ	0
28	27	SChurchill	u6nJ4H	0
29	28	JMorrison	r24umt	0
30	29	THardacre	ax8Gkf	0
31	30	Clyman	pujBRG	0

Grade Relation

The Grade relation consists of every grade stored within the system, the screenshot opposite is a good example of normalisation as it pulls data from the corresponding tables using their primary key's.

Grade ID	Student ID	Subject ID	Level ID	Grade Type	Grade	Dataset ID
2	20	7	1	2	1 D	1
3	21	35	1	2	1 D	1
4	22	46	1	2	1 D	1
5	23	89	1	2	1 D	1
6	24	102	1	2	1 D	1
7	25	115	1	2	1 D	1
8	26	133	1	2	1 D	1
9	27	141	1	2	1 D	1
10	28	165	1	2	1 D	1
11	29	170	1	2	1 D	1
12	132	29	1	2	1 C	1
13	133	66	1	2	1 C	1
14	134	72	1	2	1 C	1
15	135	82	1	2	1 C	1
16	136	167	1	2	1 C	1
17	137	172	1	2	1 C	1
18	138	208	1	2	1 C	1
19	805	8	1	2	1 B	1
20	806	18	1	2	1 B	1
21	807	19	1	2	1 B	1
22	808	28	1	2	1 B	1
23	809	30	1	2	1 B	1
24	810	36	1	2	1 B	1
25	811	49	1	2	1 B	1
26	812	56	1	2	1 B	1
27	813	57	1	2	1 B	1
28	814	58	1	2	1 B	1
29	815	59	1	2	1 B	1
30	816	67	1	2	1 B	1
31	817	68	1	2	1 B	1
32	818	71	1	2	1 B	1

Class Relation

The Class relation consists of the name of each class that is used within the system and also links each class to the subject it is a part of and the teacher that teaches it.

Class ID	Subject ID	Teacher ID	Class group	Dataset ID
2	1	1	1 Class Group 1	1
3	2	1	2 Class Group 2	1
4	3	1	3 Class Group 3	1
5	4	2	4 Class Group 4	1
6	5	2	5 Class Group 5	1
7	6	2	6 Class Group 6	1
8	7	3	7 Class Group 7	1
9	8	3	8 Class Group 8	1
10	9	5	9 Class Group 9	1
11	10	5	10 Class Group 10	1
12	11	5	11 Class Group 11	1
13	12	5	12 Class Group 12	1
14	13	5	13 Class Group 13	1
15	14	5	14 Class Group 14	1
16	15	5	15 Class Group 15	1
17	16	6	9 Class Group 16	1
18	17	6	10 Class Group 17	1
19	18	6	11 Class Group 18	1
20	19	6	12 Class Group 19	1
21	20	6	13 Class Group 20	1
22	21	6	14 Class Group 21	1
23	22	6	15 Class Group 22	1
24	23	7	16 Class Group 23	1
25	24	8	17 Class Group 24	1
26	25	8	18 Class Group 25	1

Student Group Relation

The Student Group relation is used to identify which students are within each class group.

	A	B	C	D
1	StudentGroupID	GroupID	StudentID	
2		1	1	7
3		2	1	8
4		3	1	9
5		4	1	13
6		5	1	14
7		6	1	15
8		7	1	17
9		8	1	18
10		9	1	19
11		10	1	22
12		11	1	28
13		12	1	29
14		13	1	30
15		14	1	35
16		15	1	36
17		16	1	37
18		17	1	40
19		18	1	46
20		19	1	47
21		20	1	49
22		21	1	51
23		22	1	56
24		23	1	57
25		24	1	58
26		25	1	59

Teaches Relation

The Teaches relation is used to identify which Teachers teach which class groups. The need for this table is because a single teacher can teach multiple classes in the same year for each academic year.

	A	B	C	D
1	Teaches ID	Teacher ID	Group ID	
2		1	1	1
3		2	2	2
4		3	3	3
5		4	4	4
6		5	5	5
7		6	6	6
8		7	7	7
9		8	8	8
10		9	9	9
11		10	10	10
12		11	11	11
13		12	12	12
14		13	13	13
15		14	14	14
16		15	15	15
17		16	9	16
18		17	10	17
19		18	11	18
20		19	12	19
21		20	13	20
22		21	14	21
23		22	15	22
24		23	16	23
25		24	17	24
--	--	--	--	--

Transferring these records to the SQL server

In order to transfer these records to the SQL server I had to first of all create the tables within my SQL server database with all of the correct fields correctly formatted to allow the correct data types. Then to populate these tables I needed to create SQL insert statements for the records within my Microsoft Excel tables. I did this by writing one SQL query for one row within each table and then using the “Drag” function within Microsoft Excel to do the same for all other data rows within the tables. I have included an example below:

- I first of all, wrote a SQL query for the first data row within a given table using the formula function of Microsoft Excel (highlighted below).

The screenshot shows a Microsoft Excel spreadsheet titled "sql tables - Microsoft Excel". The formula bar at the top contains the formula: `="INSERT INTO Class(ClassID, SubjectID, TeacherID, ClassGroup, DatasetID) VALUES(&A2&,&B2&,&C2&,&D2&,&E2&)"`. The spreadsheet has columns A through O. Row 1 contains headers: Class ID, Subject ID, Teacher ID, Class group, and Dataset ID. Row 2 contains data: 1, 1, 1, "Class Group 1", and 1. Below the formula bar, the formula is repeated in the cell G2: `INSERT INTO Class(ClassID, SubjectID, TeacherID, ClassGroup, DatasetID) VALUES(1,1,1,'Class Group 1',1)`.

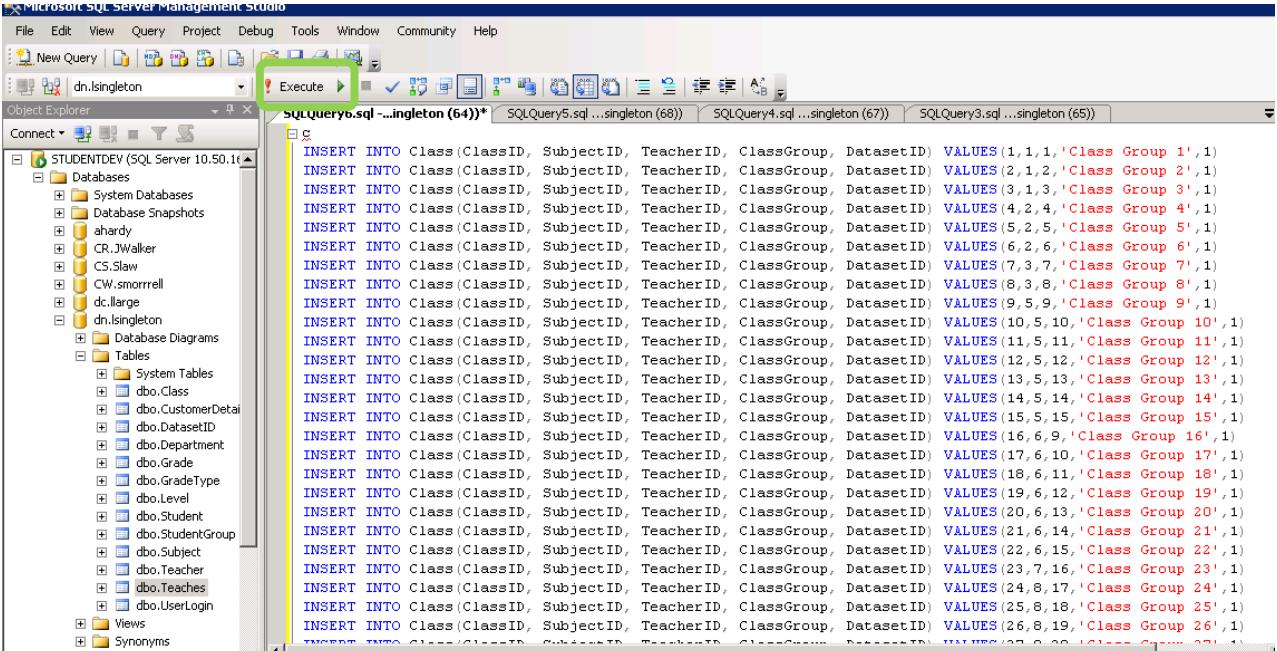
- I then used the Cell drag function to automatically generate SQL queries for every other data row within a given table.

The screenshot shows the same Microsoft Excel spreadsheet. The formula bar now contains the formula: `="INSERT INTO Class(ClassID, SubjectID, TeacherID, ClassGroup, DatasetID) VALUES(&A2&,&B2&,&C2&,&D2&,&E2&)"`. The spreadsheet now has 10 rows of data from 1 to 10. The formula in G2 has been copied down to G10, generating 10 separate SQL insert statements for each row. The entire column G (from G2 to G10) is highlighted with a green oval.

- From here I then selected the entire column of the SQL queries for a given table and copied the entire row to my clipboard.

The screenshot shows the Microsoft Excel spreadsheet again. The formula bar now contains the letter 'c'. The clipboard icon in the top left corner is highlighted with a green oval. The clipboard area shows the 10 SQL insert statements copied from the previous step. The entire column G (from G2 to G10) is highlighted with a green oval.

- 4) Then with all the SQL queries in my clipboard I paste them into the SQL Server Management Studio software within a new query. Then to finally insert them I clicked the “Execute” button (highlighted below).



The screenshot shows the Microsoft SQL Server Management Studio interface. On the left is the Object Explorer pane, which displays the database structure of 'STUDENTDEV'. In the center is the 'SQLQuery0.sql ...ingleton (64)*' query editor window. The window contains a large block of SQL code consisting of approximately 30 'INSERT INTO' statements. The 'Execute' button, located at the top of the query editor, is highlighted with a green rectangular box. The status bar at the bottom of the screen shows the text 'Centre Number: 23216'.

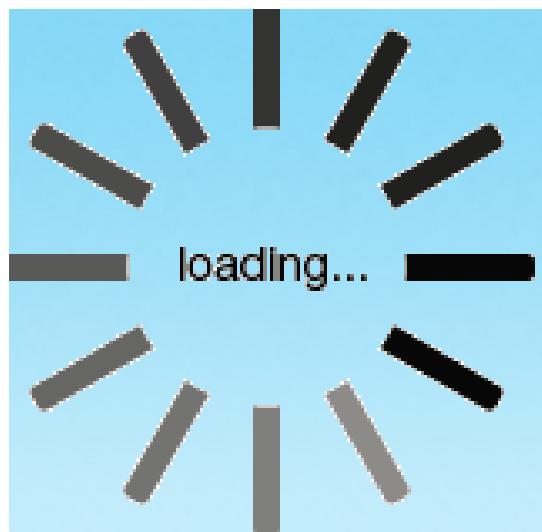
```
INSERT INTO Class(ClassID, SubjectID, TeacherID, ClassGroup, DatasetID) VALUES(1,1,1,'Class Group 1',1)
INSERT INTO Class(ClassID, SubjectID, TeacherID, ClassGroup, DatasetID) VALUES(2,1,2,'Class Group 2',1)
INSERT INTO Class(ClassID, SubjectID, TeacherID, ClassGroup, DatasetID) VALUES(3,1,3,'Class Group 3',1)
INSERT INTO Class(ClassID, SubjectID, TeacherID, ClassGroup, DatasetID) VALUES(4,2,4,'Class Group 4',1)
INSERT INTO Class(ClassID, SubjectID, TeacherID, ClassGroup, DatasetID) VALUES(5,2,5,'Class Group 5',1)
INSERT INTO Class(ClassID, SubjectID, TeacherID, ClassGroup, DatasetID) VALUES(6,2,6,'Class Group 6',1)
INSERT INTO Class(ClassID, SubjectID, TeacherID, ClassGroup, DatasetID) VALUES(7,3,7,'Class Group 7',1)
INSERT INTO Class(ClassID, SubjectID, TeacherID, ClassGroup, DatasetID) VALUES(8,3,8,'Class Group 8',1)
INSERT INTO Class(ClassID, SubjectID, TeacherID, ClassGroup, DatasetID) VALUES(9,5,9,'Class Group 9',1)
INSERT INTO Class(ClassID, SubjectID, TeacherID, ClassGroup, DatasetID) VALUES(10,5,10,'Class Group 10',1)
INSERT INTO Class(ClassID, SubjectID, TeacherID, ClassGroup, DatasetID) VALUES(11,5,11,'Class Group 11',1)
INSERT INTO Class(ClassID, SubjectID, TeacherID, ClassGroup, DatasetID) VALUES(12,5,12,'Class Group 12',1)
INSERT INTO Class(ClassID, SubjectID, TeacherID, ClassGroup, DatasetID) VALUES(13,5,13,'Class Group 13',1)
INSERT INTO Class(ClassID, SubjectID, TeacherID, ClassGroup, DatasetID) VALUES(14,5,14,'Class Group 14',1)
INSERT INTO Class(ClassID, SubjectID, TeacherID, ClassGroup, DatasetID) VALUES(15,5,15,'Class Group 15',1)
INSERT INTO Class(ClassID, SubjectID, TeacherID, ClassGroup, DatasetID) VALUES(16,6,9,'Class Group 16',1)
INSERT INTO Class(ClassID, SubjectID, TeacherID, ClassGroup, DatasetID) VALUES(17,6,10,'Class Group 17',1)
INSERT INTO Class(ClassID, SubjectID, TeacherID, ClassGroup, DatasetID) VALUES(18,6,11,'Class Group 18',1)
INSERT INTO Class(ClassID, SubjectID, TeacherID, ClassGroup, DatasetID) VALUES(19,6,12,'Class Group 19',1)
INSERT INTO Class(ClassID, SubjectID, TeacherID, ClassGroup, DatasetID) VALUES(20,6,13,'Class Group 20',1)
INSERT INTO Class(ClassID, SubjectID, TeacherID, ClassGroup, DatasetID) VALUES(21,6,14,'Class Group 21',1)
INSERT INTO Class(ClassID, SubjectID, TeacherID, ClassGroup, DatasetID) VALUES(22,6,15,'Class Group 22',1)
INSERT INTO Class(ClassID, SubjectID, TeacherID, ClassGroup, DatasetID) VALUES(23,7,16,'Class Group 23',1)
INSERT INTO Class(ClassID, SubjectID, TeacherID, ClassGroup, DatasetID) VALUES(24,8,17,'Class Group 24',1)
INSERT INTO Class(ClassID, SubjectID, TeacherID, ClassGroup, DatasetID) VALUES(25,8,18,'Class Group 25',1)
INSERT INTO Class(ClassID, SubjectID, TeacherID, ClassGroup, DatasetID) VALUES(26,8,19,'Class Group 26',1)
INSERT INTO Class(ClassID, SubjectID, TeacherID, ClassGroup, DatasetID) VALUES(27,8,20,'Class Group 27',1)
```

Form Overview

In total there are twenty-four forms that make up this project and one module. When it came to my implementation I followed the prototypes I had created in my design section to create my forms. However, during implementation I have implemented a number of design and HCI features that are different from those originally intended in my design. I believe the changing of such features has made my software solution more “user friendly” and professional.

Load Graphic

The load graphic form is used to show that a process is loading when a complex algorithm that takes a few seconds is run. The load graphic runs on a separate thread to processes within the system so that the gif animation does not freeze until the process has finished and can function at the same time as the processing of an algorithm.



Iblmsg System.Windows.Forms.Label

Load System.Windows.Forms.PictureBox

LoadGraphic System.Windows.Forms.Form

Parameter Name	Description
SetMessage	This sub receives a string message and sets it as the text value of the Iblmsg in order to store the string message.
ShowWaitScreen	This sub begins a new thread process that allows the loading graphic to be processed simultaneously.
ShowGraphic	This sub attempts to show the loading graphic to the user, and handles the disposing of the

	instance of the loading graphic form.
CloseWaitScreen	This sub ends the thread process and makes the loading graphic form disappear.

```

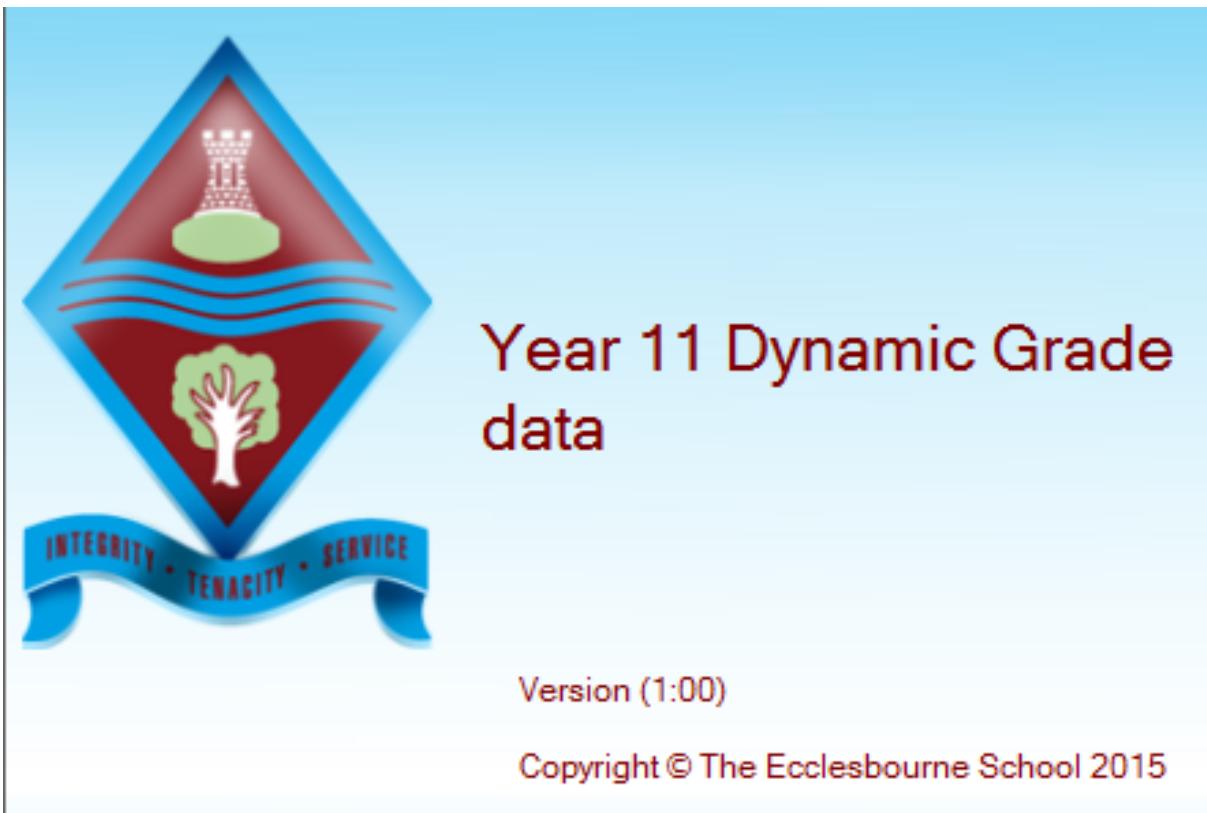
Public Class LoadGraphic
    'this public sub sets the text of the lblmsg
    Public Sub SetMessage(ByVal strMessage As String)
        lblmsg.Text = strMessage
    End Sub
End Class
Public Class clsOPAWaitScreen
    'stores a thread that can be used to run a simultaneous process
    Dim Thread1 As System.Threading.Thread

    'this sub calls the makeithappen sub and starts the thread process
    Public Sub ShowWaitScreen(ByVal strMessage As String)
        Thread1 = New System.Threading.Thread(AddressOf MakeItHappen)
        'Name is being used here to pass parameters (In this case a message)
        Thread1.Name = strMessage
        Thread1.Start()
    End Sub
    'this sub opens the thread process
    Private Sub showgraphic(ByVal strMsg As String)
        'stores a variable which is a new instance of this form
        Dim objPlsWait As New LoadGraphic
        Try
            'the thread attempts to set the message of the form and show the form
            objPlsWait.SetMessage(System.Threading.Thread.CurrentThread.Name)
            objPlsWait.ShowDialog()
        Catch ex As Exception
            ' Do nothing suppress error.
        Finally
            'Resets the objpleasewait variable and disposes of the existing objplswit
variable
            If objPlsWait IsNot Nothing Then objPlsWait.Dispose() : objPlsWait = Nothing
        End Try
    End Sub
    'this sub is called to end the thread process
    Public Sub CloseWaitScreen()
        'Makes the form go away.
        Thread1.Abort(Nothing)
    End Sub
End Class

```

SplashScreen

The splash screen loads when the system initially loads and is the barrier between the instance the system executable file is run and the login in form. The Splash Screen loads instantaneously and prevents a waiting time between the moment the system executable is run and the displaying of the login form. The Splash Screen therefore acts as a smooth visual transition, but it also displays copyright information and system version numbers.



ApplicationTitle System.Windows.Forms.Label

Copyright System.Windows.Forms.Label

DetailsLayoutPanel System.Windows.Forms.TableLayoutPanel

MainLayoutPanel System.Windows.Forms.TableLayoutPanel

picture System.Windows.Forms.PictureBox

SplashScreen System.Windows.Forms.Form

Version System.Windows.Forms.Label

Parameter Name	Description
SplashScreen_Load	When the Splash screen loads, this procedure handles the loading of the splash screen and sets the text properties of different labels.

```
Public NotInheritable Class SplashScreen
    'the code encased within this sub is performed when the SplashScreen is first loaded
    Private Sub SplashScreen_Load(ByVal sender As Object, ByVal e As System.EventArgs)
        Handles Me.Load
```

```
'Sets the title text displayed on the splashscreen
If My.Application.Info.Title <> "" Then
    ApplicationTitle.Text = My.Application.Info.Title
Else

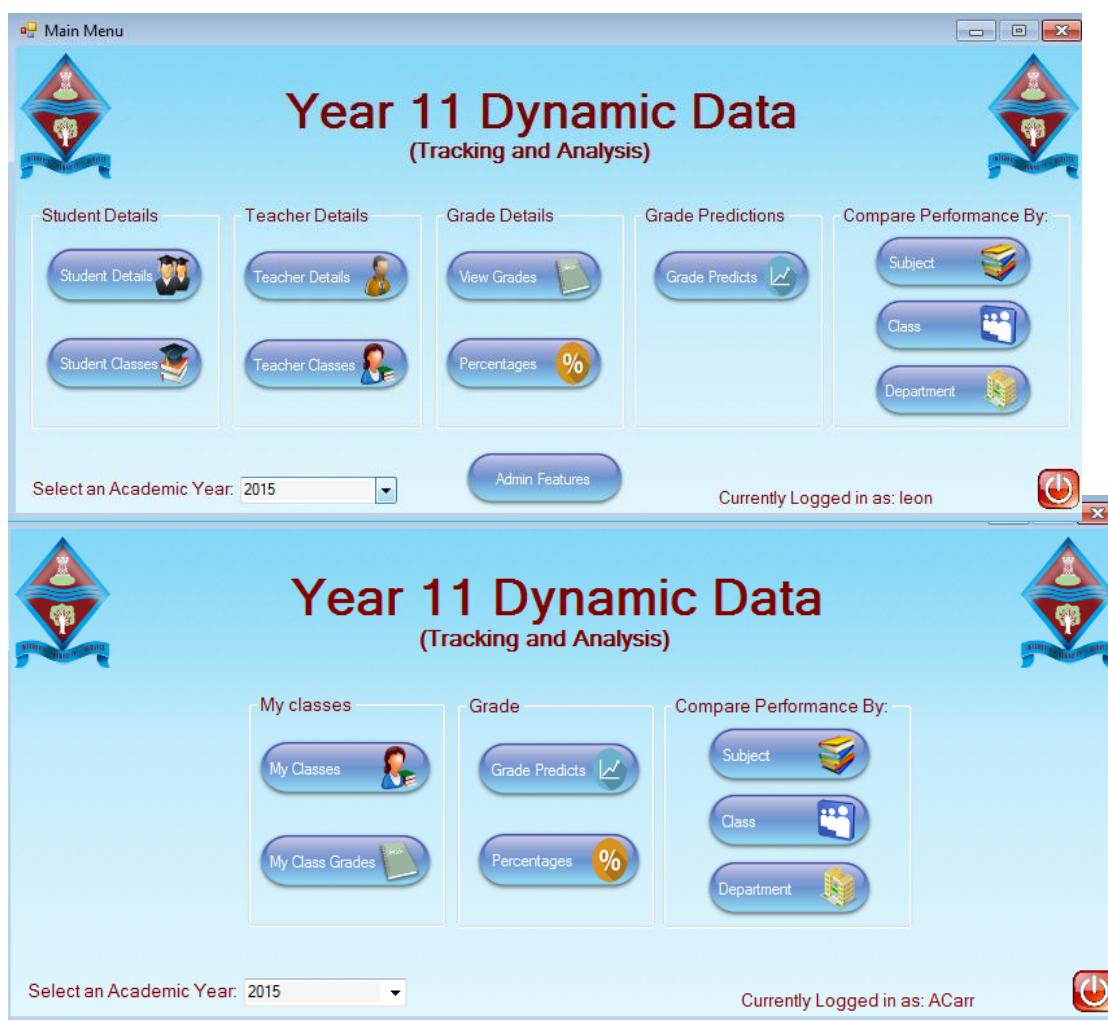
    ApplicationTitle.Text =
System.IO.Path.GetFileNameWithoutExtension(My.Application.Info.AssemblyName)
End If

'Sets the version text displayed by the splashscreen
Version.Text = System.String.Format(Version.Text,
My.Application.Info.Version.Major, My.Application.Info.Version.Minor)

'sets the copyright information displayed on the SplashScreen
Copyright.Text = My.Application.Info.Copyright
End Sub
End Class
```

Main Menu

The main menu form is the form that first appears to the user once they have logged into the system. This form acts as a homepage for the program and allows the user to access the different sections of my program; it has been structured in a manner that allows the user to navigate between sections of my program that correlate under the same category. An example of this is the 'Student Details' Group which includes buttons which will link the user to the 'Student' form and the 'Student Classes' form. The main menu form also features a logout button which allows any user that is currently logged into the system to either swap user login details or prevent potential unwanted users of the system. The main menu also features an 'Admin' button that only user's with administrative privileges can access.



```

Classes System.Windows.Forms.Button
ClassGrades System.Windows.Forms.Button
Comparegroup System.Windows.Forms.GroupBox
currentuser System.Windows.Forms.Label
Departments System.Windows.Forms.Button
Formtitle System.Windows.Forms.Label
Gradegroup System.Windows.Forms.GroupBox
Gradepercentages System.Windows.Forms.Button
gradepercentages1 System.Windows.Forms.Button
gradepredictsbutton System.Windows.Forms.Button
Logo System.Windows.Forms.PictureBox
Logo2 System.Windows.Forms.PictureBox
Logout System.Windows.Forms.Button
Main_Menu System.Windows.Forms.Form

```

Centre Number: 23216

```

myclassesbutton System.Windows.Forms.Button
Myclassesgroup System.Windows.Forms.GroupBox
Predictionsgroup System.Windows.Forms.GroupBox
StudentClasses System.Windows.Forms.Button
studentdetailsbutton System.Windows.Forms.Button
StudentGroup System.Windows.Forms.GroupBox
Subjects System.Windows.Forms.Button
Teacherclassesbutton System.Windows.Forms.Button
teacherdetailsbutton System.Windows.Forms.Button
TeacherGroup System.Windows.Forms.GroupBox
viewgradesbutton System.Windows.Forms.Button

```

Parameter Name	Description
MainMenuLoad	This parameter handles the loading of the Main menu form. It sets the current user text as the user that has logged in and it checks whether the user is an admin or not to determine if the admin button should be visible.
Logout_click	When the logout button is clicked by the user this routine verifies that the user wants to log out and if so returns to the Login form.
studentviewbutton_Click	Handles the instance the 'Student Details' button is clicked and redirects the user to the 'Student' form.
gradebutton_Click	Handles the instance the 'View Grades' button is clicked and redirects the user to the 'Grade' form.
teacherclassbutton_Click	Handles the instance the 'Teacher Classes' button is clicked and redirects the user to the 'Teacher _Classes' form.
teacherbutton_Click	Handles the instance the 'Teacher Details' button is clicked and redirects the user to the 'Teacher View' form.
Developerbutton_Click	Handles the instance the 'Admin Features' button is clicked and redirects the user to the 'Developer Features' form.
gradepercentagesbutton_Click	Handles the instance the 'Percentages' button is clicked and redirects the user to the 'Grade Percentages' form.
studentclassesbutton_Click	Handles the instance the 'Student Classes' button is clicked and redirects the user to the 'Student Classes' form.
compareSubjectsbutton_Click	Handles the instance the 'Subject' button is clicked and redirects the user to the 'Compare Subject Percentages' form.
compareDepartmentsbutton_Click	Handles the instance the 'Department' button is clicked and redirects the user to the 'Compare Department Percentages' form.
compareClassesbutton_Click	Handles the instance the 'Class' button is clicked and redirects the user to the 'Compare Class Percentages' form.
datasetcombo_SelectionChangeCommitted	Handles the instance the user makes a change to the datasetcombo box. When a change is made the moduledataset variable is set.

```

'imports the following system code commands so that I can use them appropriately
Imports System.Data.SqlClient
Imports System.Data
Imports System.Configuration
Imports System.Data.DataTable
Public Class Main_Menu
    'The following code is run when the main menu is loaded
    Private Sub MainMenuLoad() Handles Me.Load
        'Stores da as a SQL data adapter
        Dim da As New SqlDataAdapter
        'stores ds as a dataset
        Dim ds As New DataSet
        'stores the Admin ID value which determines whether the logged in user is an admin
        or a standard user
        Dim AdminID As Integer
        Dim dt As New DataTable

        'Runs the database connect sequence from the module
        DatabaseConnect()

        'Sets the text value of the Currentuser label
        currentuser.Text = ("Currently Logged in as: ")

        'If the username value which has been received from the module form is not blank
        the following sequence is run
        If moduleusername <> "" Then
            'the public sql variable is set
            sql = "Select * FROM Userlogin WHERE LoginUsername = '" & moduleusername & "'"
            AND Password = '" & modulepassword & "'"
            Dim cmmnd As New SqlCommand(sql, con)
            'the data adapter creates a connection between the program and the SQL
            database and uses the SQL statement above to query the database
            da.SelectCommand = cmmnd
            'The dataadapter fills the dataset with the result of the SQL query
            da.Fill(ds, "Login")
            'The admin ID value is set as the value of the cell that is part of the Admin
            column, of the first dataset row
            AdminID = ds.Tables(0).Rows(0).Item("Admin").ToString()
            'the currentuser label is set as the following text and the username variable
            that has been passed from the module form
            currentuser.Text = "Currently Logged in as: " & moduleusername
        Else
            'If the username value which has been received from the Login form is blank
            then the current user label text is set as follows
            currentuser.Text = ("Currently Logged in as: ")
        End If
        'if the adminID value is not 1 then the admin controls button is not visible
        If AdminID <> "1" Then
            Adminbutton.Visible = False
            TeacherGroup.Visible = False
            StudentGroup.Visible = False
            GradeGroup.Visible = False
            gradepercentages1.Visible = True
            Predictionsgroup.Text = "Grade"
            Myclassesgroup.Visible = True
            Predictionsgroup.Location = New Point(331, 123)
            Comparegroup.Location = New Point(484, 123)
        Else
            'if the adminID value is 1 then the admin controls button is visible

```

```

        Adminbutton.Visible = True
        StudentGroup.Visible = True
        TeacherGroup.Visible = True
        viewgradesbutton.Visible = True
        Myclassesgroup.Visible = False
    End If

    Dim comboindex As New Integer
    comboindex = moduleDataset

        'Uses the Generate_combo function of the module to populate a combobox by
        passing the table, field and needempty variables to the Generatecombo function and then
        returning the result in a datatable
        dt = GenerateCombo("DatasetID", "DatasetID.DatasetID,Year", True)
        'sets the datasource of the combobox
        datasetcombo.DataSource = dt
        'sets the display member of the combobox
        datasetcombo.DisplayMember = "Year"
        'sets the value member of the combobox
        datasetcombo.ValueMember = "DatasetID.DatasetID"
        datasetcombo.SelectedValue = comboindex
    End Sub

    'If the logout icon is clicked the following code is run
    Private Sub Logout_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Logout.Click
        'sets the variable result as an information messagebox with yes no options and
        with the following text
        Dim result As Integer = MessageBox.Show("Do you really want to Logout", "Logout",
        MessageBoxButtons.YesNo, MessageBoxIcon.Warning)
        'if the yes option is clicked the the login form is shown and the main menu form
        is closed
        If result = DialogResult.Yes Then
            Login.Show()
            Me.Close()
            'if the no option is cliked then nothing happens
        ElseIf result = DialogResult.No Then
            End If
        End Sub

        'if the student details button is clicked the student details form is showed and the
        main menu is hidden from the user
        Private Sub studentviewbutton_Click(ByVal sender As System.Object, ByVal e As
        System.EventArgs) Handles studentdetailsbutton.Click
            If moduleDataset <> "0" Then
                Student_View.Show()
                Me.Close()
            Else
                MsgBox("Please Select an Academic Year ", MsgBoxStyle.Information,
                MessageBoxButtons.OK)
                End If
            End Sub

            'if the view grades button is clicked the grade form is showed and the main menu is
            hidden from the user
            Private Sub gradebutton_Click(ByVal sender As System.Object, ByVal e As
            System.EventArgs) Handles viewgradesbutton.Click
                If moduleDataset <> "0" Then
                    Grade.Show()
                    Me.Close()
                Else
                    MsgBox("Please Select an Academic Year ", MsgBoxStyle.Information,
                    MessageBoxButtons.OK)
                    End If
                End Sub

```

```
'if the teacher details button is clicked the teacher class view form is showed and  
the main menu is hidden from the user  
Private Sub teacherbutton_Click(ByVal sender As System.Object, ByVal e As  
System.EventArgs) Handles teacherdetailsbutton.Click  
    TeacherView.Show()  
    Me.Close()  
End Sub  
'if the predicted grades button is clicked the predicted grades form is showed and the  
main menu is hidden from the user  
Private Sub predictedgradesbutton_Click(ByVal sender As System.Object, ByVal e As  
System.EventArgs) Handles gradepredictsbutton.Click  
    Predictions.Show()  
    Me.Close()  
End Sub  
'if the teacher classes button is clicked the teacher classes form is showed and the  
main menu is hidden from the user  
Private Sub teacherClassesbutton_Click(ByVal sender As System.Object, ByVal e As  
System.EventArgs) Handles Teacherclassesbutton.Click  
    If moduleDataset <> "0" Then  
        Teacher_Classes.Show()  
        Me.Close()  
    Else  
        MsgBox("Please Select an Academic Year ", MsgBoxStyle.Information,  
        MessageBoxButtons.OK)  
    End If  
End Sub  
'if the admin button is clicked the developer features form is showed and the main  
menu is hidden from the user  
Private Sub Developerbutton_Click(ByVal sender As System.Object, ByVal e As  
System.EventArgs) Handles Adminbutton.Click  
    Developer_Features.Show()  
    Me.Close()  
End Sub  
'if the grade percentages button is clicked the grade percentages form is showed and  
the main menu is hidden from the user  
Private Sub gradepercentagesbutton_Click(ByVal sender As System.Object, ByVal e As  
System.EventArgs) Handles Gradepercentages.Click  
    Grade_Percentages.Show()  
    Me.Close()  
End Sub  
'if the student classes button is clicked the student classes form is showed and the  
main menu is hidden from the user  
Private Sub studentclassesbutton_Click(ByVal sender As System.Object, ByVal e As  
System.EventArgs) Handles StudentClasses.Click  
    If moduleDataset <> "0" Then  
        Student_Classes.Show()  
        Me.Close()  
    Else  
        MsgBox("Please Select an Academic Year ", MsgBoxStyle.Information,  
        MessageBoxButtons.OK)  
    End If  
End Sub  
'if the subjects button is clicked the compare subject percentages form is showed and  
the main menu is hidden from the user  
Private Sub compareSubjectsbutton_Click(ByVal sender As System.Object, ByVal e As  
System.EventArgs) Handles Subjects.Click  
    Compare_Subject_Percentages.Show()  
    Me.Close()  
End Sub  
'if the classes button is clicked the compare class percentages form is showed and the  
main menu is hidden from the user
```

```
Private Sub compareClassesbutton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Classes.Click
    Compare_Class_Percentages.Show()
    Me.Close()
End Sub
'if the departments button is clicked the compare department percentages form is showed and the main menu is hidden from the user
Private Sub compareDepartmentsbutton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Departments.Click
    Compare_Department_Percentages.Show()
    Me.Close()
End Sub

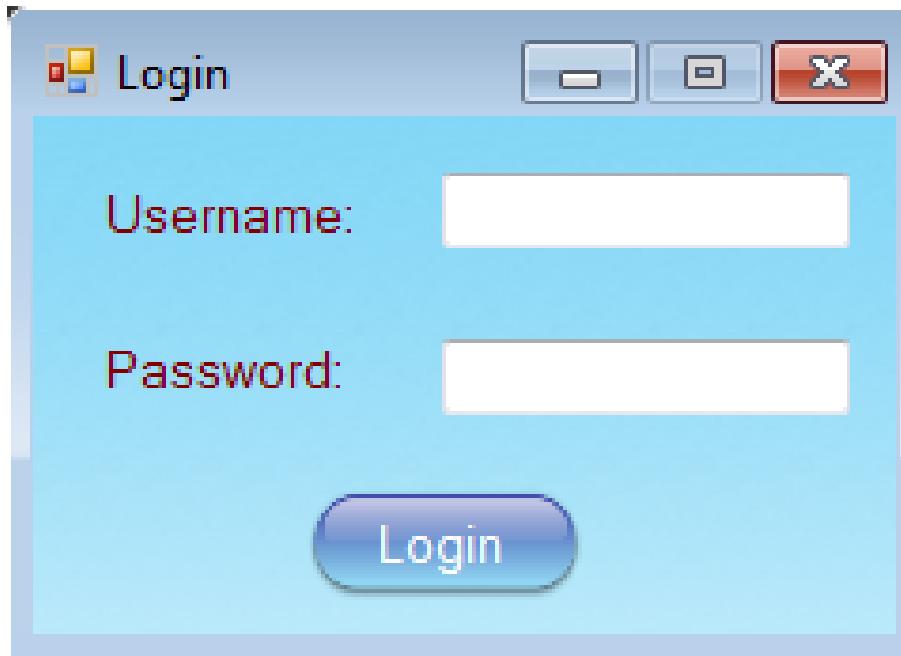
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles gradepercentages1.Click
    'if the grade percentages button is clicked the grade percentages form is showed and the main menu is hidden from the user
    Grade_Percentages.Show()
    Me.Close()
End Sub
'if the my classes button is clicked the my classes form is showed and the main menu is hidden from the user
Private Sub myclassesbutton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles myclassesbutton.Click
    If moduleDataset <> "0" Then
        Myclasses.Show()
        Me.Close()
    Else
        MsgBox("Please Select an Academic Year ", MsgBoxStyle.Information, MessageBoxButtons.OK)
        End If
    End Sub
'if the my class grades button is clicked the my class grades form is showed and the main menu is hidden from the user
Private Sub ClassGrades_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ClassGrades.Click
    If moduleDataset <> "0" Then
        MyClassGrades.Show()
        Me.Close()
    Else
        MsgBox("Please Select an Academic Year ", MsgBoxStyle.Information, MessageBoxButtons.OK)
        End If
    End Sub
Private Sub datasetcombo_SelectionChangeCommitted(ByVal sender As Object, ByVal e As System.EventArgs) Handles datasetcombo.SelectedValueChanged

    'sets the module dataset variable as the selected value of the datasetcombo

    If datasetcombo.Text = "" Then
        moduleDataset = "0"
    Else
        moduleDataset = datasetcombo.SelectedValue.ToString
    End If
End Sub
End Class
```

Login

The login form is the method of authentication in my system that restricts a person's use of the program. This form therefore loads on start-up of the system. A user must enter a valid username and password to access the system. The login form also checks whether the user has administrative privileges and should be able to access the advanced admin features. A non-valid username or password will prevent the user from accessing the system and inform them with an appropriate message.



```

Login System.Windows.Forms.Form
Loginbutton System.Windows.Forms.Button
passlabel System.Windows.Forms.Label
txtPassword System.Windows.Forms.TextBox
txtUsername System.Windows.Forms.TextBox
userlabel System.Windows.Forms.Label
  
```

Parameter Name	Description
Login_Click	This parameter handles the instance the user clicks the login button. If the user has valid login details they will be re-directed to the 'Main Menu' form otherwise they will remain on the 'login' form.

```

'imports the following system code commands so that I can use them appropriately
Imports System.Data.SqlClient
Imports System.Data
Imports System.Configuration
Imports System.Data.DataTable
Public Class Login
    'when the login button is clicked the following seqence is run to check the user and
    password login credentials
    Private Sub Login_Click(sender As System.Object, e As System.EventArgs) Handles
    Loginbutton.Click
        'stores cmd as a sqlCommand()
        Dim Cmd As New SqlCommand
        'stores da as a SQL dataadapter
        Dim da As New SqlDataAdapter
        'stores dt as a datatable
        Dim dt As New DataTable
        'stores ds as a dataset
        Dim ds As New DataSet

        'sets stored variables for the username and password as strings
        Dim password As String
        Dim Username As String
        'Stores the connection variable as an sqlConnection that can create a link between
        the program and the SQL server
        Dim con As New SqlConnection
        'stores the sql command that is sued to query the SQL Server Database
        Dim sqlquery As New SqlCommand
        'A second username variable is required so that the stored username can be swapped
        later
        Dim username1 As String

        'This is the connection path that the SQLconection variable follows, using the
        following ID and password to connect
        con.ConnectionString = "Data Source=<>;Initial Catalog=<>;User ID=<>;Password=<>"

        Try
            'Tries to connect to the SQL Database.
            con.Open()
        Catch ex As Exception
            'If the SQL database can not been found the following messagebox displays,
            preventing the program from crashing otherwise
            MsgBox("Can not open SQL Server connection! ")
        End Try

        'Sets the SQL command query of the connection with the following SQL statement
        sqlquery.Connection = con
        sqlquery.CommandText = "SELECT LoginUsername, Password FROM UserLogin WHERE
        LoginUsername = '" & txtUsername.Text & "' AND Password = '" & txtPassword.Text & "'"

        'Creates a stored varieble using a SQL data reader to check if there is a row
        returned when the SQL statement is carried out
        Dim check As SqlDataReader = sqlquery.ExecuteReader()

        If check.HasRows Then
            While check.Read()

                'checks if the username and password in the text fields match a record
                from the Database table
                password = check("Password").ToString()
                Username = check("LoginUsername").ToString()

```

```
'Swaps the username variables
If Username = check("LoginUsername") Then
    username1 = Username
End If

'sets the module usernames and passwords
modulepassword = txtPassword.Text
moduleusername = txtUsername.Text

'displays the main menu form and hides the login from
Main_Menu.Show()
Me.Close()

End While
'closes the open connection string and data adapter
con.Close()
check.Dispose()
'opens a new database connection
DatabaseConnect()
'sets the sql statement
sql = "SELECT DISTINCT * FROM UserLogin WHERE LoginUsername = '" &
moduleusername & "' AND Password = '" & modulepassword & "'"
'sets the connection path as the connectionstring of con from the module
Cmd.Connection = con
'sets the command statement as the SQL statement
Cmd.CommandText = sql
'the dataadapter then uses this SQL command to Query the database
da.SelectCommand = Cmd
'the dataadapter retrieves the query results from the SQL statement after
connecting to the Database and fills a dataset
da.Fill(ds, "temp")
'the datatable variable is then set to the table result of the dataset
variable
    moduleuserid = ds.Tables(0).Rows(0).Item("LoginID").ToString()

Else
    'If the username and password do not match a valid login from the SQL Database
    'table then the following message is displayed to the user
    MessageBox.Show("Invalid Username or Password", "Authentication Failure",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
    'resets the text of the username and text properties to blank text
    txtPassword.Text = ""
    txtUsername.Text = ""

End If
End Sub
End Class
```

Student

The Student form allows a user to search students via a number of different filters such as ethnicity, gender etc. The students that match the user's choice of filter then display in a data grid which lists all of a student's personal details. Once a user has a selection of students they can then edit a student's details by clicking on the button that corresponds to the record the user wants to edit. From this form a user can also choose whether they want to add a new student to the system, this will prompt a dialog pop-up which allows the user to add a new student record. A user can also print this form by clicking the print icon, allowing a user to have a hard-copy of a student selection.

The screenshot shows a Windows application window titled "Student". The main title bar has "Main Menu" and "Student". Below the title bar is a toolbar with a green Android icon. The main area is titled "Search Student Details". On the left, there is a sidebar with the following controls:

- "Select an attribute:" dropdown set to "Form".
- "Form:" dropdown set to "12A".
- "Search" button.
- "Add New Student" button.
- "Import CSV" section with "Add a new Dataset Year" button.
- "Select an Academic Year:" dropdown.
- "Import CSV file" button.

On the right is a "DataGridView" showing student data:

Firstname	Surname	Gender	FSM	SNS	Ethnicity	Attendance	DateOfBirth	Form	Year	Student Details
Hamzah	Finlay	M	N	N	White - British	98.3	01/07/1999	12A	12	Edit Details
Benoit	Hamzah	M	N	N	Pakistani	96.7	05/12/1998	12A	12	Edit Details
Andrew	Emma	M	N	K	White - British	100	26/06/1999	12A	12	Edit Details
Matthew	Holly	M	N	N	White - British	96.7	27/05/1999	12A	12	Edit Details
Tom	Jess	M	N	N	White and Asian	90	01/01/1999	12A	12	Edit Details
Ben	Kate	M	N	N	White - British	100	24/02/1999	12A	12	Edit Details
George	Leonie	M	N	N	White - British	100	09/12/1998	12A	12	Edit Details
Ben	Maisie	M	N	N	White - British	100	18/04/1999	12A	12	Edit Details
Will	William	M	N	N	White - British	98.3	01/09/1998	12A	12	Edit Details
Alexander	Christopher	M	N	N	White - British	100	26/07/1999	12A	12	Edit Details
Eve	Peter	F	N	N	White - British	91.7	23/08/1999	12A	12	Edit Details
Millie	Natalie	F	N	N	White - British	100	26/01/1999	12A	12	Edit Details
Jasmine	Megan	F	N	N	White - British	100	12/07/1999	12A	12	Edit Details
Debbie	Rebecca	F	N	N	White - British	93.3	15/05/1999	12A	12	Edit Details
Katie	Isabella	F	N	N	White - British	98.3	02/06/1999	12A	12	Edit Details
Leah	Lauren	F	N	N	White - British	100	26/05/1999	12A	12	Edit Details
Rebecca	Tristan	F	N	N	White - British	100	26/07/1999	12A	12	Edit Details

```

Addstudent System.Windows.Forms.Button
Banner System.Windows.Forms.PictureBox
Displaystudent System.Windows.Forms.Button
Field System.Windows.Forms.ComboBox
Fieldlabel System.Windows.Forms.Label
formtitle System.Windows.Forms.Label
linkarrow System.Windows.Forms.Label
MainMenu System.Windows.Forms.LinkLabel
printbutton System.Windows.Forms.Button
PrintDialog System.Windows.Forms.PrintDialog
PrintForm Microsoft.VisualBasic.PowerPacks.Printing.PrintForm
SearchDataset System.Windows.Forms.ComboBox
SearchEthnicity System.Windows.Forms.ComboBox
SearchForm System.Windows.Forms.ComboBox
SearchGender System.Windows.Forms.ComboBox
SearchName System.Windows.Forms.TextBox
SearchSNS System.Windows.Forms.ComboBox
SearchYear System.Windows.Forms.ComboBox
Student System.Windows.Forms.LinkLabel
Student_View System.Windows.Forms.Form
StudentGrid System.Windows.Forms.DataGridView
studentgroup System.Windows.Forms.GroupBox
wherelabel System.Windows.Forms.Label

```

Description

student_search_view	This parameter handles the instance the form loads; it is responsible for sizing the form and connecting to the SQL server database.
StudentGrid_CellClick	Handles the instance a cell that is part of the data grid is clicked, and in this case opens an instance of the 'Edit student' form.
Field_SelectedIndexChanged	This procedure handles every instance the filter field is changed by the user, and appropriately displays the correct filter selection depending on the user's filtering choice.
MainMenu_LinkClicked	Handles the instance the main menu link label is clicked and re-directs the user to the 'Main Menu' form.
printbutton_Click	Handles the instance the print icon is clicked by the user. A print dialog box is then displayed to the user allowing a user to select where to print a copy of the form to amongst other settings such as number of copies etc.
Addstudent_Click	Handles the instance the 'Add New Student' button is clicked and redirects the user to the 'Edit Student' form.
importcsv_Click	Handles the instance a user clicks the Import CSV button, when clicked the user is prompted to open a CSV file. This CSV file is then written to the database with all its student values inserted. The rows that were added from the CSV file are then shown in a data grid view. A true/false column is attached to indicate whether the row has been added successfully to the database.

```

Imports System.Data.SqlClient
Imports System.Data
Imports System.Data.OleDb
Imports System
Imports System.IO
Public Class Student_View
    Public Property AutoSizeColumnsMode As DataGridViewAutoSizeColumnsMode
        'This sub is performed when the form initially loads
        Private Sub student_search_view(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Me.Load
            Dim dt As New DataTable

            'Calls the database connect sub from the module
            DatabaseConnect()
            'sets the screenarea of the form so that it is displayed center screen
            Me.Left = (Screen.PrimaryScreen.WorkingArea.Width - Me.Width) / 2
            Me.Top = (Screen.PrimaryScreen.WorkingArea.Height - Me.Height) / 2
            'studentgrid initially is set to not be visible
            StudentGrid.Visible = False
            addnewdataset = False
            'Uses the Generate_combo function of the module to populate a combobox by passing
            the table, field and needempty variables to the Generatecombo function and then returning
            the result in a datatable
            dt = GenerateCombo("DatasetID", "DatasetID.DatasetID,Year", True)
            'sets the datasource of the combobox
            academicyearcombo.DataSource = dt
            'sets the display member of the combobox
            academicyearcombo.DisplayMember = "Year"
            'sets the value member of the combobox
            academicyearcombo.ValueMember = "DatasetID.DatasetID"
        End Sub
        'When the search button is clicked the code within the following sub is run
        Private Sub Displaystudent_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Displaystudent.Click
            'clears all columns from the studentgrid
            StudentGrid.Columns.Clear()
            'stores a dataset variable
            Dim ds As DataSet

            'sets the SQL statement
            sql = " Select StudentID, Firstname, Surname, Gender, FSM, SNS, Ethnicity,
Attendance, DateOfBirth, Form, Year FROM Student "
            'This Case statement is used to alter the SQL query depending on which text is
            selected from the combobox, when a case is selected the above SQL statement and the
            corresponding case 'SQL where statement' are concatenated to form one SQL statement
            Select Case Field.Text
                Case "Name"
                    sql = sql & " WHERE (Firstname LIKE '%" & SearchName.Text & "%' Or Surname
                    LIKE '%" & SearchName.Text & "%') and DatasetID = '" & moduleDataset & "' ORDER BY
                    StudentID ASC "
                Case "Gender"
                    sql = sql & " WHERE Gender LIKE '%" & SearchGender.Text & "%' and
                    DatasetID = '" & moduleDataset & "' ORDER BY StudentID ASC"
                Case "Ethnicity"
                    sql = sql & " WHERE Ethnicity LIKE '%" & SearchEthnicity.Text & "%' and
                    DatasetID = '" & moduleDataset & "' ORDER BY StudentID ASC"
                Case "SNS"
                    sql = sql & " WHERE SNS LIKE '%" & SearchSNS.Text & "%' and DatasetID = '" &
                    moduleDataset & "' ORDER BY StudentID ASC"
                Case "Form"
                    sql = sql & " WHERE Form LIKE '%" & SearchForm.Text & "%' and DatasetID =
                    '" & moduleDataset & "' ORDER BY StudentID ASC"
            End Case
        End Sub
    End Class

```

```

        Case "Year"
            sql = sql & " WHERE Year LIKE '%" & SearchYear.Text & "%' and DatasetID =
            ' & moduleDataset & "' ORDER BY StudentID ASC"
        End Select

        'the dataset is set after passing the following SQL statement and table into the
        GenerateDataset sub of the public module
        ds = generateDataset(sql, "Student")
        'sets the datasource of the datagrid
        StudentGrid.DataSource = ds
        'sets the datagrid as visible
        StudentGrid.Visible = True
        'sets the datamember of the datagrid
        StudentGrid.DataMember = "Student"
        'hides the StudentID column from the user
        StudentGrid.Columns("StudentID").Visible = False

        'creates a new variable that stores information for a button column that can be
        added to the datagrid
        Dim Column As New DataGridViewButtonColumn
        With Column
            'sets the header text of the button column
            .HeaderText = "Student Details"
            'sets the name of the button column
            .Name = "Details"
            'sets the text displayed on the buttons of the button column in the datagrid
            .Text = "Edit Details"
            .UseColumnTextForButtonValue = True
        End With

        'adds the button column to the datagrid
        StudentGrid.Columns.Add(Column)
        'auto resizes the column widths of the datagrid
        StudentGrid.AutoResizeColumns()

        'if there is only one row in the datagrid then the entered username is not a valid
        username and so the following error message is displayed to the user and the grid is not
        visible
        If (StudentGrid.Rows.Count = 0) Then
            MsgBox("Please enter a valid name")
            StudentGrid.Visible = False
        End If
    End Sub
    'when a button that is part of the button column is clicked, the code within the
    following sub is run
    Private Sub StudentGrid_CellClick(ByVal sender As System.Object, ByVal e As
    System.Windows.Forms.DataGridViewCellEventArgs) Handles StudentGrid.CellClick
        'stores the studentID
        Dim StudentId As Integer

        If e.ColumnIndex = 11 Then
            Add_Edit_Student.Close()
            'sets the value of the StudentID as the value of the cell that is on the same
            row as the Edit Details button that is clicked
            StudentId = StudentGrid.Rows(e.RowIndex).Cells("StudentID").Value

            'sets the student ID of the edit student form as the studentID of this form
            Add_Edit_Student.StudentID = StudentId
            'show the editstudentform and closes this form
            Add_Edit_Student.Show()
        End If
    End Sub

```

```
End Sub
'when the text within the field combobox is changed then the code within this sub is
run
Private Sub Field_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Field.SelectedIndexChanged
    'stores dt as a datatable
    Dim dt As New DataTable

    'sets the wherelabel and datagrids as visible to the user
    Displaystudent.Visible = True
    wherelabel.Visible = True

    'This case statement selects which textbox or combobox is visible when the
    corresponding text value of the field combobox is selected and sets all other textboxes or
    comboboxes as not visible when a case is selected by the user's mouse click choice
    Select Case Field.Text
        Case ""
            SearchGender.Visible = False
            SearchName.Visible = False
            SearchEthnicity.Visible = False
            SearchSNS.Visible = False
            SearchForm.Visible = False
            SearchYear.Visible = False
            SearchDataset.Visible = False
            Displaystudent.Visible = False
            wherelabel.Visible = False
        Case "Name"
            SearchGender.Visible = False
            SearchName.Visible = True
            SearchEthnicity.Visible = False
            SearchSNS.Visible = False
            SearchForm.Visible = False
            SearchYear.Visible = False
            SearchDataset.Visible = False
            wherelabel.Text = "Name:"
        Case "Gender"
            'Uses the Generate combo function of the module to populate a combobox by
            passing the table, field and needempty variables to the Generatecombo function and then
            returning the result in a datatable
            dt = GenerateCombo("Student", "Gender", False)
            'sets the datasource of the combobox
            SearchGender.DataSource = dt
            'sets the display member of the combobox
            SearchGender.DisplayMember = "Gender"

            SearchGender.Visible = True
            SearchName.Visible = False
            SearchEthnicity.Visible = False
            SearchSNS.Visible = False
            SearchForm.Visible = False
            SearchYear.Visible = False
            SearchDataset.Visible = False
            wherelabel.Text = "Gender:"
        Case "Ethnicity"
            dt = GenerateCombo("Student", "Ethnicity", False)
            SearchEthnicity.DataSource = dt
            SearchEthnicity.DisplayMember = "Ethnicity"

            SearchGender.Visible = False
            SearchName.Visible = False
            SearchEthnicity.Visible = True
            SearchSNS.Visible = False
```

```

        SearchForm.Visible = False
        SearchYear.Visible = False
        SearchDataset.Visible = False
        wherelabel.Text = "Ethnicity:"
    Case "SNS"
        dt = GenerateCombo("Student", "SNS", False)
        SearchSNS.DataSource = dt
        SearchSNS.DisplayMember = "SNS"

        SearchGender.Visible = False
        SearchName.Visible = False
        SearchEthnicity.Visible = False
        SearchSNS.Visible = True
        SearchForm.Visible = False
        SearchYear.Visible = False
        SearchDataset.Visible = False
        wherelabel.Text = "SNS:"
    Case "Form"
        'selects the forms based on the user's choice of year
        sql = "SELECT Distinct Form FROM Student WHERE DatasetID = '" &
moduleDataset & "'"
        'stores a SQL command that uses the SQL statement to query the database
        'that is found when following the connection string of con
        Using comm As SqlCommand = New SqlCommand(sql, con)
            'stores a SQL dataadareader which can read the contents of the sql
            'query from the database
            Dim rs As SqlDataReader = comm.ExecuteReader
            'stores a datatable
            'the datatable is filled with the values that the datareader reads
            dt.Load(rs)

            'sets the displaymemeber of the combobox
            SearchForm.DisplayMember = "Form"
            'sets the datasource of the combobox
            SearchForm.DataSource = dt
        End Using

        SearchGender.Visible = False
        SearchName.Visible = False
        SearchEthnicity.Visible = False
        SearchSNS.Visible = False
        SearchForm.Visible = True
        SearchYear.Visible = False
        SearchDataset.Visible = False
        wherelabel.Text = "Form:"
    Case "Year"
        dt = GenerateCombo("Student", "Year", False)
        SearchYear.DataSource = dt
        SearchYear.DisplayMember = "Year"

        SearchGender.Visible = False
        SearchName.Visible = False
        SearchEthnicity.Visible = False
        SearchSNS.Visible = False
        SearchForm.Visible = False
        SearchYear.Visible = True
        SearchDataset.Visible = False
        wherelabel.Text = "Year:"
    End Select
End Sub
'When the main menu label is clicked the main menu form is showed to the user and this
form closes

```

```

Private Sub MainMenu_LinkClicked(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles MainMenu.LinkClicked
    Main_Menu.Show()
    Add_Edit_Student.Close()
    Me.Close()
End Sub
' deals with the print form button
Private Sub printbutton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles printbutton.Click
    ' sets the printer settings to default
    PrintForm.PrinterSettings = PrintDialog.PrinterSettings
    If PrintDialog.ShowDialog() = DialogResult.OK Then
        ' allows the user to change print settings
        PrintForm.PrinterSettings = PrintDialog.PrinterSettings
        ' changes the orientation of the printed document to landscape
        Me.PrintForm.PrinterSettings.DefaultPageSettings.Landscape = True
        ' refreshes the form so that the printdialog is not printed along with the form
        Me.Refresh()
        ' prints the current form
        Me.PrintForm.Print()
        ' Percentageschart.Printing.Print(True)
    End If
End Sub
'this sub handles the add student button
Private Sub Addstudent_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Addstudent.Click
    ' opens the add student form
    Add_Edit_Student.Show()
End Sub
'this sub is responsible for importing a CSV file and writing its contents into the
database
Private Sub importcsv_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles importcsv.Click
    ' checks whether the user has selected a value in the academicyearcombo box
    If academicyearcombo.SelectedValue <> "0" Then
        Dim dlg As New OpenFileDialog
        Dim ds As New DataSet
        Dim path As String = "My Filename"
        Dim path1 As String
        Dim dt As New DataTable
        Dim dt1 As New DataTable
        Dim griddt As New DataTable
        Dim rowadded As String
        Dim da As New SqlDataAdapter
        ' stores a StudentID that will be used as the Studentid to be inserted in the
database
        Dim Sid As Integer

        ' filters the file selection the user can attempt to open in the open dialog
box
        dlg.Filter = "CSV Files|*.csv|Any Files|*.*"

        If dlg.ShowDialog = System.Windows.Forms.DialogResult.OK Then
            ' sets the file path of the CSV file the user wants to import as the
location they have selected from the open file dialog
            path = System.IO.Path.GetDirectoryName(path)
            path = dlg.FileName
            path1 = dlg.FileName
            path = System.IO.Path.GetDirectoryName(path)
            path1 = System.IO.Path.GetFileName(path1)
        End If
    End If
End Sub

```

```

'sets a connection string up between the system and the selected CSV file
Dim strConnString As String = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=" & path & ";Extended Properties=Text;"
Dim conn As New OleDbConnection(strConnString)
'opens the connection
conn.Open()

'selects all contents from the CSV file using the above connection string
Dim cmd As New OleDbCommand("SELECT * FROM [" & path1 & "]", conn)
Dim da1 As New OleDbDataAdapter()

'fills the griddt datatable with the selected contents of the CSV file
da1.SelectCommand = cmd
da1.Fill(griddt)
'disposes of the data adapter and closes the connection between the system
and the CSV file
da1.Dispose()
conn.Close()

'adds a new column to the griddt datagrid
griddt.Columns.Add("Inserted (True/False)")

Try
    Dim x As Integer = 0
    Dim csvrows As Integer
    'stores a variable as the number of rows in the griddt datagrid
    csvrows = griddt.Rows.Count

    'sets the SQL statement
    sql = "SELECT TOP 1 StudentID FROM Student ORDER BY StudentID DESC"
    'stores cmmnd as a SQL command that uses the above SQL statement and
    the connection string of con
    Dim cmmnd As New SqlCommand(sql, con)
    'sets the dataadapter select command as cmmnd
    da.SelectCommand = cmmnd
    'the dataadapter retrieves the query results from the SQL statement
    after connecting to the Database and fills a dataset
    da.Fill(ds, "Student")
    'The StudentID is then set as the value of the highest StudentID found
    from the Query and then incremented by 1, the SQL statement takes into account deleting
    teachers so that inserting a Student will use the first available StudentID
    Sid = ds.Tables(0).Rows(0).Item("StudentID").ToString() + 1

    While (x <= csvrows - 1)
        'stores variables as the number of rows before and after inserting
        a new record
        Dim originaldatabaserows As Integer
        Dim afterinsertdatabaserows As Integer

        'sets the SQL statement
        sql = "SELECT * From Student"
        Dim sqlcommand As New SqlCommand(sql, con)
        'the dataadapter then uses this SQL command to Query the database
        da.SelectCommand = sqlcommand
        'the dataadapter retrieves the query results from the SQL
        statement after connecting to the Database and fills a dataset
        da.Fill(ds, "temp")
        'the datatable variable is then set to the table result of the
        dataset variable
        dt = ds.Tables("temp")
        'the original rows variable is set as the number of rows returned

```

```

originaldatabaserows = dt.Rows.Count

'executes an insert statement using the following parameters,
which depend on the row index x within the griddt datatable, the dataset selected by the
user and the studentID.
sql = "INSERT INTO Student VALUES(@SID, @Fname, @Sname, @Gender,
@FSM, @SNS, @Ethnicity, @Attendance, @DOB, @Form, @Year, @DatasetID) SET ANSI_WARNINGS
OFF"
Using cmd1 = New SqlCommand(sql, con)
    cmd1.Parameters.AddWithValue("SID", Sid)
    cmd1.Parameters.AddWithValue("Fname", griddt.Rows(x)(0))
    cmd1.Parameters.AddWithValue("Sname", griddt.Rows(x)(1))
    cmd1.Parameters.AddWithValue("Gender", griddt.Rows(x)(2))
    cmd1.Parameters.AddWithValue("FSM", griddt.Rows(x)(3))
    cmd1.Parameters.AddWithValue("SNS", griddt.Rows(x)(4))
    cmd1.Parameters.AddWithValue("Ethnicity", griddt.Rows(x)(5))
    cmd1.Parameters.AddWithValue("Attendance", dt.Rows(x)(6))
    cmd1.Parameters.AddWithValue("DOB", griddt.Rows(x)(7))
    cmd1.Parameters.AddWithValue("Form", griddt.Rows(x)(8))
    cmd1.Parameters.AddWithValue("Year", griddt.Rows(x)(9))
    cmd1.Parameters.AddWithValue("DatasetID",
academicyearcombo.SelectedValue)
    'executes the insert query
    cmd1.ExecuteNonQuery()
End Using

'sets the after insert variable as the new number of rows
sql = "SELECT * From Student"
Dim sqlcommand2 As New SqlCommand(sql, con)
da.SelectCommand = sqlcommand2
da.Fill(ds, "temp1")
dt1 = ds.Tables("temp1")
afterinsertdatabaserows = dt1.Rows.Count

'sets the string property of the rowadded variable if the result
of the following calculation equals 1 + the row index
If (afterinsertdatabaserows) - (originaldatabaserows) = (1 + x)
Then
    rowadded = "True"
Else
    rowadded = "False"
End If

'adds this result to the Griddt datatable
griddt.Rows(x)( "Inserted (True/False)" ) = rowadded

'increments the studentID and row indexes by 1
x = x + 1
Sid = Sid + 1
End While
'informs the user the insert has been successful
MsgBox("Insert Data Successful", MsgBoxStyle.Information, "Data
Upload")
dt = Nothing
'catches any errors during an insert and displays this to the user

'sets the datasource of the datagrid
StudentGrid.DataSource = griddt
'sets the datagrid as visible
StudentGrid.Visible = True
'sets the datamember of the datagrid
StudentGrid.DataMember = ""

```

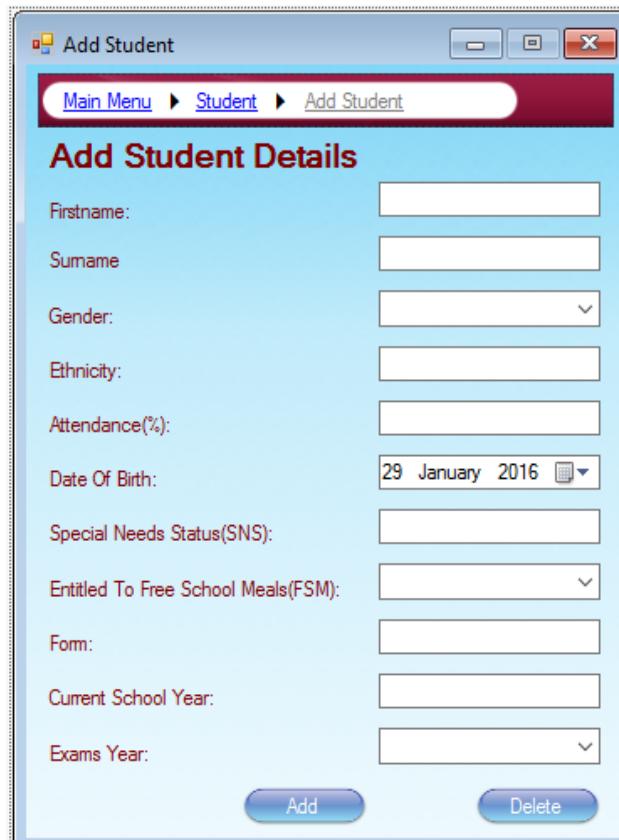
```
'auto resizes all columns in the datagridview
StudentGrid.AutoResizeColumns()
Catch ex As Exception
    MsgBox(ex.Message)
End Try

End If
Else
    MsgBox("Please Select an Academic Year ", MsgBoxStyle.Information,
MessageBoxButtons.OK)
    End If
End Sub

Private Sub addnewdataset_Click(sender As System.Object, e As System.EventArgs)
Handles newdataset.Click
    addnewdataset = True
    Developer_Features.Show()
    Me.Close()
End Sub
End Class
```

Edit/Add Student

The Edit/Add student form allows a user to edit a Student's record or add a completely new Student record into the database. The form detects if there is a StudentID passed into it, if there is then the form automatically displays the 'Edit' features and the button uses an update SQL command to update the student's record that matches the student's ID that has passed into the form. If there is no StudentID passed into the form then the form displays the 'Add' features and the button uses an insert SQL statement that creates a new record within the database using the values the user has entered into the textboxes and combo boxes. A new ID is dynamically generated to identify the new Student record uniquely.



```

Add_Edit System.Windows.Forms.Button
Add_Edit_Student System.Windows.Forms.Form
AddDetail System.Windows.Forms.Label
Attendance System.Windows.Forms.TextBox
AttendanceLabel System.Windows.Forms.Label
banner System.Windows.Forms.PictureBox
Dataset System.Windows.Forms.ComboBox
DatasetLabel System.Windows.Forms.Label
DateOfBirth System.Windows.Forms.DateTimePicker
Delete System.Windows.Forms.Button
DOBLabel System.Windows.Forms.Label
editLabel System.Windows.Forms.LinkLabel
Ethnicity System.Windows.Forms.TextBox
EthnicityLabel System.Windows.Forms.Label
Firstname System.Windows.Forms.TextBox
FnameLabel System.Windows.Forms.Label
Form System.Windows.Forms.TextBox
FormLabel System.Windows.Forms.Label
Free_School_Meals System.Windows.Forms.ComboBox
FSMLabel System.Windows.Forms.Label
Gender System.Windows.Forms.ComboBox
GenderLabel System.Windows.Forms.Label
linkarrow System.Windows.Forms.Label
linkarrow2 System.Windows.Forms.Label
MainMenu System.Windows.Forms.LinkLabel
SnameLabel System.Windows.Forms.Label
SNS System.Windows.Forms.TextBox
SNSLabel System.Windows.Forms.Label
Student System.Windows.Forms.LinkLabel
Surname System.Windows.Forms.TextBox

```

Parameter Name	Description
_StudentID	This sub uses a get command to fetch the corresponding StudentID of the row the user wants to edit and stores it as a private integer, which will be used to reference the correct row in the database.
student_search_view	This sub runs when the form initially loads and dynamically generates the combo boxes on the form using their corresponding variables from the SQL server. It also checks whether there has been a StudentID passed to the form and if so sets the textbox and combo box values as the values corresponding to the record the user wants to edit.
editedcheck	This function sets a true or false condition depending on whether the values in any of the entry boxes differ from their original values. If they differ then a false state is returned.
txtvalidate	This function sets a true or false condition depending on whether any textboxes on the form are blank. It also sets the background colour of textboxes appropriately to indicate to the user that fields have not been completed.
IDexists	This function uses the StudentID that has been passed into the form to check whether a record with that StudentID exists in the database.
rowexists	This function checks the database to see if the record that the user wants to delete actually exists.
Edit_Click	This sub dynamically creates a new Student ID and determines whether to use an update or insert statement depending on the state of the IDexists function. If the ID does not exist then a new record is inserted with the dynamically created ID.
Delete_Click	When the delete button is clicked by the user, this sub checks if the row the user wants to delete actually exists. If it does exist then a SQL statement removes the record from the database.
Student_LinkClicked	When this link is clicked the user is re-directed to the student view form, but if values have been edited on the form then the user is asked if they really want to change forms.
MainMenu_LinkClicked	When this link is clicked the user is also asked if they really want to change forms if they have edited values but otherwise they are immediately re-directed to the main menu form.

```

Imports the following system code commands so that I can use them appropriately
Imports System.Data.SqlClient
Imports System.Data
Imports System.Configuration
Imports System.Data.DataTable
Public Class Add_Edit_Student
    'sets the variables that are received from the student form as private variables which
    can be used in any sub within this class
    Private _StudentID As Integer
    Dim SID As Integer

    'sets the values of several variables that can be used to check the values of these
    variables later in the code, they can be used in any sub
    Dim Fnamecheck As String
    Dim Snamecheck As String
    Dim Gendercheck As String
    Dim Ethnicitycheck As String
    Dim attendancheck As String
    Dim DOBcheck As String
    Dim SNScheck As String
    Dim FSMcheck As String
    Dim Formcheck As String
    Dim Yearcheck As String
    Dim DatasetCheck As String
    'creates a property of the variable that is being received
    Public Property StudentID() As Integer
        Get
            'gets the variable from the form that has been specified, in this case the
            student form
            Return _StudentID
        End Get
        Set(ByVal value As Integer)
            'sets the variable that has been received by the get statement as the
            equivalent variable that will be used in this form
            _StudentID = value
        End Set
    End Property
    'This sub runs when the Edit Student form is initially loaded
    Private Sub student_search_view(ByVal sender As System.Object, ByVal e As
        System.EventArgs) Handles Me.Load

        'sets the screen dimensions of the form to automatically generate center screen of
        the users display
        Me.Left = (Screen.PrimaryScreen.WorkingArea.Width - Me.Width) / 2
        Me.Top = (Screen.PrimaryScreen.WorkingArea.Height - Me.Height) / 2
        'calls database connect from the public module
        DatabaseConnect()
        'stores dt as a datatable
        Dim dt As DataTable

        'Uses the Generate combo function of the module to populate a combo box by passing
        the table, field and needempty variables to the Generatecombo function and then returning
        the result in a datatable
        dt = GenerateCombo("Student", "Gender", True)
        'sets the datasource of the combobox
        Gender.DataSource = dt
        'sets the display member of the combobox
        Gender.DisplayMember = "Gender"

        dt = GenerateCombo("Student", "FSM", False)
        Free_School_Meals.DataSource = dt
        Free_School_Meals.DisplayMember = "FSM"
    End Sub

```

```

dt = GenerateCombo("DatasetID", "DatasetID.DatasetID,Year", True)
Dataset.DataSource = dt
Dataset.DisplayMember = "Year"
Dataset.ValueMember = "DatasetID.DatasetID"
'stores ds as a dataset
Dim ds As DataSet

'If there is a Student ID then the following is altered
If StudentID Then
    'all text properties are changed to edit rather than add

    Datasetlabel.Visible = False
    Dataset.Visible = False
    Add_Edit.Text = "Edit"
    Add_Edit.Visible = True
    Delete.Visible = True
    editlabel.Text = "Edit Student"
    Adddetail.Text = "Edit Student Details"
    Dataset.Text = "aa"
    Me.Text = "Edit Student"
    'the dataset is set as the returned dataset after passing StudentID in to the
    studenteditvalues function in the module
    ds = StudentEditValues(StudentID)
    'the following variables are set to their equivalent values under their
    respective headings in the dataset
    Fnamecheck = ds.Tables(0).Rows(0).Item("Firstname").ToString()
    Snamecheck = ds.Tables(0).Rows(0).Item("Surname").ToString()
    Formcheck = ds.Tables(0).Rows(0).Item("Form").ToString()
    Gendercheck = ds.Tables(0).Rows(0).Item("Gender").ToString()
    Ethnicitycheck = ds.Tables(0).Rows(0).Item("Ethnicity").ToString()
    attendancheck = ds.Tables(0).Rows(0).Item("Attendance").ToString()
    SNScheck = ds.Tables(0).Rows(0).Item("SNS").ToString()
    Yearcheck = ds.Tables(0).Rows(0).Item("Year").ToString()
    FSMcheck = ds.Tables(0).Rows(0).Item("FSM").ToString()
    'sets the format for the date of birth textbox
    DateOfBirth.CustomFormat = "yyyy-MM-dd"
    DOBcheck = ds.Tables(0).Rows(0).Item("DateOfBirth").ToString()

    'sets the text of the textboxes as the values of the variables above that have
    been set
    Firstname.Text = Fnamecheck
    Surname.Text = Snamecheck
    Form.Text = Formcheck
    Ethnicity.Text = Ethnicitycheck
    Gender.Text = Gendercheck
    Attendance.Text = attendancheck
    SNS.Text = SNScheck
    Free_School_Meals.Text = FSMcheck
    Year.Text = Yearcheck
    Dataset.Text = DatasetCheck
    DateOfBirth.Text = DOBcheck
End If
End Sub
'this function returns a true or false state depending on whether conditions within
the function have been met or not
Public Function editedcheck() As Boolean
    Call DatabaseConnect()
    'initially the functions state is set to true
    editedcheck = True

```

```

'but if any of the following textbox fields values are not the same as their
respective stored variables then the functions state is set to false
If Firstname.Text <> Fnamecheck Then
    editedcheck = False
End If
If Surname.Text <> Snamecheck Then
    editedcheck = False
End If
If Gender.Text <> Gendercheck Then
    editedcheck = False
End If
If Ethnicity.Text <> Ethnicitycheck Then
    editedcheck = False
End If
If Attendance.Text <> attendancheck Then
    editedcheck = False
End If
If DateOfBirth.Text <> DOBcheck Then
    editedcheck = False
End If
If Free_School_Meals.Text <> FSMcheck Then
    editedcheck = False
End If
If SNS.Text <> SNScheck Then
    editedcheck = False
End If
If Form.Text <> Formcheck Then
    editedcheck = False
End If
If Year.Text <> Yearcheck Then
    editedcheck = False
End If
If Dataset.Text <> DatasetCheck Then
    editedcheck = False
End If
End Function
'this function returns a true or false state depending on whether conditions within
the function have been met or not
Public Function txtvalidate() As Boolean
    Call DatabaseConnect()
    'initially the functions state is set to true
    txtvalidate = True

    'if any of the textboxes are left blank then the function returns a false state
    and the background of the textbox field changes to red, otherwise the background of the
    textbox changes to white
    If Firstname.Text = "" Then
        Firstname.BackColor = Color.Salmon
        txtvalidate = False
    Else
        Firstname.BackColor = Color.White
    End If
    If Surname.Text = "" Then
        Surname.BackColor = Color.Salmon
        txtvalidate = False
    Else
        Surname.BackColor = Color.White
    End If
    If Gender.Text = "" Then
        Gender.BackColor = Color.Salmon
        txtvalidate = False
    Else

```

```

        Gender.BackColor = Color.White
    End If
    If Ethnicity.Text = "" Then
        Ethnicity.BackColor = Color.Salmon
        txtvalidate = False
    Else
        Ethnicity.BackColor = Color.White
    End If
    If Attendance.Text = "" Then
        Attendance.BackColor = Color.Salmon
        txtvalidate = False
    Else
        Attendance.BackColor = Color.White
    End If
    If SNS.Text = "" Then
        SNS.BackColor = Color.Salmon
        txtvalidate = False
    Else
        SNS.BackColor = Color.White
    End If
    If Free_School_Meals.Text = "" Then
        Free_School_Meals.BackColor = Color.Salmon
        txtvalidate = False
    Else
        Free_School_Meals.BackColor = Color.White
    End If
    If Form.Text = "" Then
        Form.BackColor = Color.Salmon
        txtvalidate = False
    Else
        Form.BackColor = Color.White
    End If
    If Year.Text = "" Then
        Year.BackColor = Color.Salmon
        txtvalidate = False
    Else
        Year.BackColor = Color.White
    End If
    If Dataset.Text = "" Then
        Dataset.BackColor = Color.Salmon
        txtvalidate = False
    Else
        Dataset.BackColor = Color.White
    End If
End Function
'this function returns a true or false state depending on whether the TeacherID exists
Public Function IDexists(ByVal studentID As String) As Boolean
    'stores cmd as a sqlCommand
    Dim Cmd As New SqlCommand
    'stores da as a SQL dataadapter
    Dim da As New SqlDataAdapter
    'stores dt as a datatable
    Dim dt As New DataTable
    'stores ds as a dataset
    Dim ds As New DataSet
    'connects to the SQL database by calling the DatabaseConnect from the module
    Call DatabaseConnect()

    'sets the SQL statement
    sql = "Select * from Student where StudentID = " & studentID & ""
    'sets the connection path as the connectionstring of con from the module
    Cmd.Connection = con

```

```

'sets the command statement as the SQL statement
Cmd.CommandText = sql
'the dataadapter then uses this SQL command to Query the database
da.SelectCommand = Cmd
'the dataadapter retrieves the query results from the SQL statement after
connecting to the Database and fills a dataset
da.Fill(ds, "Student")
'the datatable variable is then set to the table result of the dataset variable
dt = ds.Tables("Student")
'if the number of rows in the datatable is one or more then the IDexists state is
set to true otherwise it is set to false
If dt.Rows.Count >= 1 Then
    IDexists = True
Else
    IDexists = False
End If
End Function
>this function returns a true or false state depending on whether the row exists in
the database, and works in a similar manner as the function seen above
Public Function rowexists() As Boolean
    Dim Cmd As New SqlCommand
    Dim da As New SqlDataAdapter
    Dim dt As New DataTable
    Dim ds As New DataSet

    Call DatabaseConnect()
    sql = "SELECT DISTINCT Student.Gender, Student.Attendance, Student.Ethnicity,
Student.Firstname, Student.Surname FROM Student WHERE Student.Firstname = '" &
Firstname.Text & "' AND Student.Surname = '" & Surname.Text & "' AND Student.Gender = '" &
Gender.Text & "' AND Student.Ethnicity = '" & Ethnicity.Text & "' AND Student.Attendance =
'" & Attendance.Text & "';"
    Dim cmmnd As New SqlCommand(sql, con)
    da.SelectCommand = cmmnd
    da.Fill(ds, "temp")
    dt = ds.Tables("temp")
    If dt.Rows.Count >= 1 Then
        rowexists = True
    Else
        rowexists = False
    End If
End Function
>this sub runs when the add_edit button is clicked by the user
Private Sub Edit_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Add_Edit.Click

    Dim da As New SqlDataAdapter
    Dim dt As New DataTable
    Dim ds As New DataSet
    'stores a StudentID that will be used as the teacherID to be inserted in the
database
    Dim Sid As Integer
    'sets the SQL satatement
    sql = "SELECT TOP 1 StudentID FROM Student ORDER BY StudentID DESC"
    'stores cmmnd as a SQL command that uses the above SQL statement and the connection
string of con
    Dim cmmnd As New SqlCommand(sql, con)
    'sets the dataadapter select command as cmmnd
    da.SelectCommand = cmmnd
    'the dataadapter retrieves the query results from the SQL statement after
connecting to the Database and fills a dataset
    da.Fill(ds, "Student")
    'the datatable variable is then set to the table result of the dataset variable

```

```

dt = ds.Tables("Student")
'The StudentID is then set as the value of the highest StudentID found from the
Query and then incremented by 1, the SQL statement takes into account deleting teachers so
that inserting a Student will use the first available StudentID
Sid = ds.Tables(0).Rows(0).Item("StudentID").ToString() + 1

DatabaseConnect()
'if the textboxes have text and have been validated the program continues
If txtvalidate() = True Then
    'if the StudentID does not already exist then the program continues with an
insert statement
    If IDexists(StudentID) = False Then
        'sets the SQL statement
        sql = "INSERT INTO STUDENT(StudentID, Firstname, Surname, Gender, FSM,
SNS, Ethnicity, Attendance, DateOfBirth, Form, Year, DatasetID) VALUES (@StudentID,
@Fname, @Surname, @Gender, @Meals, @SNS, @Ethnicity, @Attendance, @Birth, @Form, @Year,
@Dataset)"
        'uses cmd as a new SQL command that uses the above SQL statement and the
connection string of con
        Using cmd = New SqlCommand(sql, con)
            'sets the parameters of the values to be inserted
            cmd.Parameters.AddWithValue("Fname", Firstname.Text)
            cmd.Parameters.AddWithValue("Surname", Surname.Text)
            cmd.Parameters.AddWithValue("Gender", Gender.Text)
            cmd.Parameters.AddWithValue("Meals", Free_School_Meals.Text)
            cmd.Parameters.AddWithValue("SNS", SNS.Text)
            cmd.Parameters.AddWithValue("Ethnicity", Ethnicity.Text)
            cmd.Parameters.AddWithValue("Attendance", Attendance.Text)
            cmd.Parameters.AddWithValue("Birth", DateOfBirth.Text)
            cmd.Parameters.AddWithValue("Form", Form.Text)
            cmd.Parameters.AddWithValue("Year", Year.Text)
            cmd.Parameters.AddWithValue("Dataset", Dataset.SelectedValue)
            cmd.Parameters.AddWithValue("StudentID", Sid)
            'executes the SQL command
            cmd.ExecuteNonQuery()
        End Using
        'closes the database connection
        con.Close()
        'notifies the user with the appropriate message
        MsgBox("Record Added to Database", MsgBoxStyle.Information)
        'closes this form and shows the Student view form
        Student_View.Show()
        Me.Close()
    Else
        sql = "UPDATE Student SET Firstname = @Fname, Surname = @Surname, Gender =
@Gender, FSM = @Meals, SNS = @SNS, Ethnicity = @Ethnicity, Attendance = @Attendance,
DateOfBirth = @Birth, Form = @Form, Year = @Year WHERE StudentID = " & StudentID & ";"

        Using cmd = New SqlCommand(sql, con)
            'sets the parameters of the values to be inserted
            cmd.Parameters.AddWithValue("Fname", Firstname.Text)
            cmd.Parameters.AddWithValue("Surname", Surname.Text)
            cmd.Parameters.AddWithValue("Gender", Gender.Text)
            cmd.Parameters.AddWithValue("Meals", Free_School_Meals.Text)
            cmd.Parameters.AddWithValue("SNS", SNS.Text)
            cmd.Parameters.AddWithValue("Ethnicity", Ethnicity.Text)
            cmd.Parameters.AddWithValue("Attendance", Attendance.Text)
            cmd.Parameters.AddWithValue("Birth", DateOfBirth.Text)
            cmd.Parameters.AddWithValue("Form", Form.Text)
            cmd.Parameters.AddWithValue("Year", Year.Text)
            cmd.ExecuteNonQuery()
        End Using
    End If
End Sub

```

```

        con.Close()
        MsgBox("Record Updated", MsgBoxStyle.Information)
    End If
Else
    'if the textvalidate has been returned as false then the following error
message displays to the user
    MsgBox("One or more textboxes have not been completed",
MsgBoxStyle.Information, MessageBoxButtons.OK)
End If
End Sub
'this sub runs when the delete button is clicked and works in a similar manner to the
code in the above sub
Private Sub Delete_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Delete.Click
    Dim da As New SqlDataAdapter
    Dim dt As New DataTable
    Dim ds As New DataSet

    DatabaseConnect()
    'checks if the record that the user wants to delete exists in the database
    If rowexists() = True Then
        'this block of code sets the StudentID of the record that the user wants to
delete based on the values of the firstname and surname textboxes
        sql = "Select * FROM Student WHERE Firstname = '" & Firstname.Text & "' AND
Surname = '" & Surname.Text & "'"
        Dim cmmnd As New SqlCommand(sql, con)
        da.SelectCommand = cmmnd
        da.Fill(ds, "Student")
        dt = ds.Tables("Student")
        SID = ds.Tables(0).Rows(0).Item("StudentID").ToString()
        'provides a yes/no dialog box asking the user if they really want to delete
the record
        Dim result As Integer = MessageBox.Show("Do you really want to delete this
record?", "Delete", MessageBoxButtons.YesNo, MessageBoxIcon.Warning)
        'if the answer is yes then the following code deletes the record from the
database using the SQL statement below
        If result = DialogResult.Yes Then
            sql = "DELETE FROM Student WHERE Student.StudentID = '" & SID & "'"
            Using cmd = New SqlCommand(sql, con)
                cmd.ExecuteNonQuery()
            End Using
            con.Close()
            MsgBox("Record Deleted From Database", MsgBoxStyle.Information)
            Student_View.Show()
            Me.Close()
            'but if the answer is no nothing happens
        ElseIf result = DialogResult.No Then
        End If
    Else
        ' if the record that the user wants to delete does not exist in the database,
the following message is displayed
        MsgBox("The Entered Student does not exist", MsgBoxStyle.Information,
MessageBoxButtons.OK)
    End If
End Sub
'This sub runs when the Teacher link is clicked
Private Sub Student_LinkClicked(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles Student.LinkClicked
    'the following forms either display or close if edited check is true
    If editedcheck() Then
        Student_View.Show()
        Me.Close()
    End If
End Sub

```

```
'if edited check is not true then a dialog message displays to the user with a
yes/no option
    ElseIf editedcheck() = False Then
        Dim result As Integer = MessageBox.Show("Do you really want to change
window?", "Edited Record", MessageBoxButtons.YesNo, MessageBoxIcon.Warning)
            'if the user clicks yes then the following forms either display or close
            If result = DialogResult.Yes Then
                Student_View.Show()
                Me.Close()
                'if the user clicks no then nothing happens
            ElseIf result = DialogResult.No Then
                End If
            End If
    End Sub
'This sub runs when the main menu link is clicked and functions in the same manner as
the teacher link above
Private Sub MainMenu_LinkClicked(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles MainMenu.LinkClicked
    If editedcheck() Then
        Main_Menu.Show()
        Student_View.Close()
        Me.Close()
    ElseIf editedcheck() = False Then
        Dim result As Integer = MessageBox.Show("Do you really want to change
window?", "Edited Record", MessageBoxButtons.YesNo, MessageBoxIcon.Warning)
        If result = DialogResult.Yes Then
            Main_Menu.Show()
            Student_View.Close()
            Me.Close()
        ElseIf result = DialogResult.No Then
            End If
        End If
    End Sub
End Class
```

TeacherView

The Teacher View form can be used by the user to display all the names of Teachers that teach at the school. This list of Teachers can be filtered by name and uses a SQL like statement that returns all names that match the criteria the user enters into the search field. Once a user has searched a list of teachers they can then click on a button cell that is part of the data grid, this will allow the user to edit teacher values. A new teacher can also be added to the database from this form, if the 'Add Teacher' button is clicked then a form appears which will allow the user to enter the first name and surname of a new teacher they want to add to the system. A print button in the top right corner also allows this form to be printed so that a hard-copy of the search criteria can be produced.

Firstname	Surname	Teacher Details
Amelia	Lamberts	Edit Details
Amelia	Rutherford	Edit Details
Andrea	Carr	Edit Details
Andrew	Thomson	Edit Details
Anna	Miller	Edit Details
Audrey	Rutherford	Edit Details
Ava	Butler	Edit Details
Bella	Burgess	Edit Details
Benjamin	Knox	Edit Details
Boris	Wilson	Edit Details
Cameron	May	Edit Details
Cameron	Wallace	Edit Details
Carol	Fisher	Edit Details
Carolyn	Roberts	Edit Details
Charles	Carr	Edit Details
Christian	Lyman	Edit Details

```

banner System.Windows.Forms.PictureBox
DisplayTeacher System.Windows.Forms.Button
Field System.Windows.Forms.ComboBox
Fieldlabel System.Windows.Forms.Label
Formheader System.Windows.Forms.Label
Linkarrow System.Windows.Forms.Label
MainMenu System.Windows.Forms.LinkLabel
printbutton System.Windows.Forms.Button
PrintDialog System.Windows.Forms.PrintDialog
PrintForm Microsoft.VisualBasic.PowerPacks.Printing.PrintForm
SearchName System.Windows.Forms.TextBox
Student System.Windows.Forms.LinkLabel
TeacherAdd System.Windows.Forms.Button
TeacherGrid System.Windows.Forms.DataGridView
TeacherGroup System.Windows.Forms.GroupBox
TeacherView System.Windows.Forms.Form
wherelabel System.Windows.Forms.Label

```

Parameter Name	Description
teacher_class_view	When the form loads this sub connects to the SQL database and sets the form dimensions so that they fit centre screen.
DisplayTeacher_Click	This sub runs when the Search button is clicked and returns a query matching the criteria the user has used to filter their teacher selection by, this selection is then displayed in the Teacher data grid.
TeacherGrid_CellClick	When a cell that is part of the Teacher grid is clicked then the Edit teacher form displays, the TeacherID that is on the row of the data grid button that is clicked is also passed to the edit teacher form.
Field_SelectedIndexChanged	Whenever the text value of the filter field is changed this sub determines which objects are either visible or non-visible.
MainMenu_LinkClicked	When this link is clicked the main menu is displayed.
TeacherAdd_Click	When this button is clicked the Add teacher form is displayed.
printbutton_Click	This sub allows the user to print a hard copy of the form when the print icon is clicked, additional print dialog options are then displayed allowing the user to choose printer and number of copies amongst other settings.

```

Public Class TeacherView
    'This sub is performed when the form initially loads
    Private Sub teacher_class_view(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Me.Load
        'Calls the database connect sub from the module
        DatabaseConnect()
        'teachergrid initially is set to not be visible
        TeacherGrid.Visible = False
        'Sets the screenarea of the form so that it is displayed center screen
        Me.Left = (Screen.PrimaryScreen.WorkingArea.Width - Me.Width) / 2
        Me.Top = (Screen.PrimaryScreen.WorkingArea.Height - Me.Height) / 2
    End Sub
    'When the search button is clicked the code within the following sub is run
    Private Sub DisplayTeacher_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles DisplayTeacher.Click
        'clears all columns from the teachergrid
        TeacherGrid.Columns.Clear()
        'stores a dataset variable
        Dim ds As DataSet

        'sets the SQL statement
        sql = "SELECT TeacherID, Firstname, Surname FROM Teacher"
        'This Case statement is used to alter the SQL query depending on which text is
        selected from the combobox, when a case is selected the above SQL statement and the
        corresponding case 'SQL where statement' are concatenated to form one SQL statement
        Select Case Field.Text
            Case ""
                sql = sql & "ORDER BY Firstname ASC, Surname ASC"
            Case "Name"
                sql = sql & " WHERE Firstname LIKE '%" & SearchName.Text & "%' Or Surname
                LIKE '%' & SearchName.Text & "%' ORDER BY Firstname ASC, Surname ASC"
        End Select

        'The dataset is set after passing the following SQL statement and table into the
        GenerateDataset sub of the public module
        ds = generateDataset(sql, "Teacher")
        'sets the datasource of the datagrid
        TeacherGrid.DataSource = ds
        'sets the datagrid as visible
        TeacherGrid.Visible = True
        'sets the datamember of the datagrid
        TeacherGrid.DataMember = "Teacher"
        'sets the teacherID column to not visible
        TeacherGrid.Columns("TeacherID").Visible = False

        'Creates a new variable that stores information for a button column that can be
        added to the datagrid
        Dim Column As New DataGridViewButtonColumn
        With Column
            'sets the header text of the button column
            .HeaderText = "Teacher Details"
            'sets the name of the button column
            .Name = "Details"
            'sets the text displayed on the buttons of the button column
            .Text = "Edit Details"
            .UseColumnTextForButtonValue = True
        End With

        'adds the button column to the datagrid
        TeacherGrid.Columns.Add(Column)

        'auto resizes the column widths of the datagrid
    End Sub

```

```

TeacherGrid.AutoResizeColumns()

'if there is only one row in the datagrid then the entered username is not a valid
username and so the following error message is displayed to the user and the grid is not
visible
If (TeacherGrid.Rows.Count = 0) Then
    MsgBox("Please enter a valid name")
    TeacherGrid.Visible = False
End If

End Sub
'when a button that is part of the button column is clicked, the code within the
following sub is run
Private Sub TeacherGrid_CellClick(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.DataGridViewCellEventArgs) Handles TeacherGrid.CellClick
    'stores the TeacherID
    Dim TeacherID As String

    If e.ColumnIndex = 3 Then
        EditAdd_Teacher.Close()
        'sets the value of the TeacherID as the value of the cell that is on the same
        row as the Edit Details button that is clicked
        TeacherID = TeacherGrid.Rows(e.RowIndex).Cells("TeacherID").Value

        'sets the Teacher ID of the edit Teacher form as the TeacherID of this form
        EditAdd_Teacher.TeacherID = TeacherID
        'show the editTeacher and closes this form
        EditAdd_Teacher.Show()
    End If
End Sub
'when the text within the field combobox is changed then the code within this sub is
run
Private Sub Field_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Field.SelectedIndexChanged
    'sets the wherelabel and datagrids as visible to the user
    DisplayTeacher.Visible = True
    wherelabel.Visible = True
    SearchName.Visible = True
    'This case statement decides what to do if the field.text value is blank, if it is
    then the following features are set to not visible
    Select Case Field.Text
        Case ""
            DisplayTeacher.Visible = False
            wherelabel.Visible = False
            SearchName.Visible = False
    End Select
End Sub
'When the main menu label is clicked the main menu form is showed to the user and this
form closes
Private Sub MainMenu_LinkClicked(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles MainMenu.LinkClicked
    Main_Menu.Show()
    EditAdd_Teacher.Close()
    Me.Close()
End Sub
'if the add teacher button is clicked by the user then the edit teacher form is
displayed to the user
Private Sub TeacherAdd_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles TeacherAdd.Click
    EditAdd_Teacher.Show()
End Sub
'deals with the print form button

```

```
Private Sub printbutton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles printbutton.Click
    'sets the printer settings to default
    PrintForm.PrinterSettings = PrintDialog.PrinterSettings
    If PrintDialog.ShowDialog() = DialogResult.OK Then
        'allows the user to change print settings
        PrintForm.PrinterSettings = PrintDialog.PrinterSettings
        'changes the orientation of the printed document to landscape
        Me.PrintForm.PrinterSettings.DefaultPageSettings.Landscape = True
        'refreshes the form so that the printdialog is not printed along with the form
        Me.Refresh()
        'prints the current form
        Me.PrintForm.Print()
    End If
End Sub
End Class
```

Edit/Add Teacher

This form allows a new teacher record to be added to the database, or an existing teacher record to be edited and then updated within the database. If a teacherID is passed into this form then the form displays the 'editing' features otherwise the form displays the 'add' features (can be seen in the screenshots below). From this form an existing teacher's record can also be deleted from the database. The 'edit' and 'add' button uses the user's values of the textbox fields to determine the values to either insert or update within the database.

Add Teacher

Main Menu > Teacher > Edit Teacher

Edit Teacher Details

Firstname:

Surname:

Edit **Delete**

Add Teacher

Main Menu > Teacher > Add Teacher

Add Teacher Details

Firstname:

Surname:

Add

Add_Edit System.Windows.Forms.Button
addeditlabel System.Windows.Forms.LinkLabel
AddT System.Windows.Forms.Label
Banner System.Windows.Forms.PictureBox
Delete System.Windows.Forms.Button
Edit_Teacher System.Windows.Forms.Form
Firstname System.Windows.Forms.TextBox
Fnamelabel System.Windows.Forms.Label
Linkarrow System.Windows.Forms.Label
Linkarrow2 System.Windows.Forms.Label
MainMenu System.Windows.Forms.LinkLabel
Snamelabel System.Windows.Forms.Label
Surname System.Windows.Forms.TextBox
Teacher System.Windows.Forms.LinkLabel
 Centre Number: 23216

Parameter Name	Description
_TeacherID	This sub uses a get command to fetch the corresponding TeacherID of the row the user wants to edit and stores it as a private integer, which will be used to reference the correct row in the database.
Edit_teacher	This code encased within this sub initiates when the form loads. A dataset variable is filled with the corresponding teacher edit values of the TeacherID that is passed to the form. Also if there is a TeacherID passed to the form then the textboxes display the values of first name and surname of the teacher the user wants to edit. But if there is not a TeacherID then the user is instructed to add a teacher.
teacher_LinkClicked	When this link label is clicked the user is redirected to the teacher form. If fields on the form have been edited then the user is asked to confirm their navigation choice.
MainMenu_LinkClicked	When this link label is clicked the user is redirected to the Main menu form. If fields on the form have been edited then the user is asked to confirm their navigation choice.
editedcheck	This function sets a true or false condition depending on whether the values in any of the entry boxes differ from their original values. If they differ then a false state is returned.
txtvalidate	This function sets a true or false condition depending on whether any textboxes on the form are blank. It also sets the background colour of textboxes appropriately to indicate to the user that fields have not been completed.
IDexists	This function uses the TeacherID that has been passed into the form to check whether a record with that TeacherID exists in the database.
rowexists	This function checks the database to see if the record that the user wants to delete actually exists within the database.
Edit_Click	This sub dynamically creates a new Teacher ID and determines whether to use an update or insert statement depending on the state of the IDexists function. If the ID does not exist then a new record is inserted with the dynamically created ID.
Delete_Click	When the delete button is clicked by the user, and the row exists, then a SQL statement removes the record from the database.

```

Imports the following system code commands so that I can use them appropriately
Imports System.Data.SqlClient
Imports System.Data
Imports System.Configuration
Imports System.Data.DataTable

Public Class EditAdd_Teacher
    'sets the variables that are received from the teacher form as private variables which
    can be used in any sub within this class
    Private _TeacherID As Integer
    Dim TID As Integer

    'sets two variables, one for firstname and one for surname which can be used with any
    sub to check the value of these variables
    Dim Fnamecheck As String
    Dim Snamecheck As String
    'creates a property of the variable that is being received
    Public Property TeacherID() As Integer
        Get
            'gets the variable from the form that has been specified, in this case the
            Teacher form
            Return _TeacherID
        End Get
        Set(ByVal value As Integer)
            'sets the variable that has been received by the get statement as the
            equivalent variable that will be used in this form
            _TeacherID = value
        End Set
    End Property
    'this sub runs when the Edit Teacher form initially loads
    Private Sub Edit_teacher(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Me.Load
    'sets the screen dimensions of the form to automatically generate center screen of
    the users display
    Me.Left = (Screen.PrimaryScreen.WorkingArea.Width - Me.Width) / 2
    Me.Top = (Screen.PrimaryScreen.WorkingArea.Height - Me.Height) / 2
    'calls database connect from the public module
    DatabaseConnect()
    'stores ds as a dataset
    Dim ds As DataSet

    'the dataset is set as the dataset that is returned when passing the TeacherID
    into the TeacherEditvalues function in the public module
    ds = TeacherEditValues(TeacherID)

    'If there is a teacher ID then the following is altered
    If TeacherID Then
        'delete button becomes visible
        Delete.Visible = True

        'all text properties are changed to edit rather than add
        AddT.Text = "Edit Teacher Details"
        Add_Edit.Text = "Edit"
        addeditlabel.Text = "Edit Teacher"
        'the dataset is set as the returned dataset after passing TeacherID in to the
        teachereditvalues function in the module
        ds = TeacherEditValues(TeacherID)
        'fnamecheck and snamecheck variables are set to their equivalent values in the
        dataset under their respective column headings
        Fnamecheck = ds.Tables(0).Rows(0).Item("Firstname").ToString()
        Snamecheck = ds.Tables(0).Rows(0).Item("Surname").ToString()
    End If
End Sub

```

```

        'sets the text of the textboxes as the values of the variables fnamecheck and
        snamecheck
        F firstname.Text = F fnamecheck
        Surname.Text = S namecheck
    End If

End Sub
'this sub runs when the Teacher link is clicked
Private Sub teacher_LinkClicked(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles Teacher.LinkClicked
    'the following forms either display or close if edited check is true
    If editedcheck() Then
        Teacher.Show()
        Me.Close()
        'if edited check is not true then a dialog message displays to the user with a
        yes/no option
    ElseIf editedcheck() = False Then
        Dim result As Integer = MessageBox.Show("Do you really want to change
        window?", "Edited Record", MessageBoxButtons.YesNo, MessageBoxIcon.Warning)
        'if the user clicks yes then the following forms either display or close
        If result = DialogResult.Yes Then
            Teacher.Show()
            Me.Close()
            'if the user clicks no then nothing happens
        ElseIf result = DialogResult.No Then
            End If
        End If
    End If
End Sub
'this sub runs when the main menu link is clicked and functions in the same manner as
the teacher link above
Private Sub MainMenu_LinkClicked(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles MainMenu.LinkClicked
    If editedcheck() Then
        Main_Menu.Show()
        TeacherView.Close()
        Me.Close()
    ElseIf editedcheck() = False Then
        Dim result As Integer = MessageBox.Show("Do you really want to change
        window?", "Edited Record", MessageBoxButtons.YesNo, MessageBoxIcon.Warning)
        If result = DialogResult.Yes Then
            Main_Menu.Show()
            TeacherView.Close()
            Me.Close()
        ElseIf result = DialogResult.No Then
            End If
        End If
    End Sub
'this function returns a true or false state depending on whether conditions within
the function have been met or not
Public Function editedcheck() As Boolean
    Call DatabaseConnect()
    'initially the functions state is set to true
    editedcheck = True
    'but if either the firstname or textbox fields is not the same value as the two
    variables fnamecheck or snamecheck then the functions state is set to false
    If F firstname.Text <> F fnamecheck Then
        editedcheck = False
    End If
    If Surname.Text <> S namecheck Then
        editedcheck = False
    End If
End Function

```

```

'this function returns a true or false state depending on whether conditions within
the function have been met or not
Public Function txtvalidate() As Boolean
    Call DatabaseConnect()
    'initially the functions state is set to true
    txtvalidate = True
    'if any of the textboxes are left blank then the function returns a false state
    and the background of the textbox field changes to red, otherwise the background of the
    textbox changes to white
    If Firstname.Text = "" Then
        Firstname.BackColor = Color.Salmon
        txtvalidate = False
    Else
        Firstname.BackColor = Color.White
    End If
    If Surname.Text = "" Then
        Surname.BackColor = Color.Salmon
        txtvalidate = False
    Else
        Surname.BackColor = Color.White
    End If
End Function
'this function returns a true or false state depending on whether the TeacherID exists
Public Function IDexists(ByVal TeacherID As String) As Boolean
    'stores cmd as a sqlCommand
    Dim Cmd As New SqlCommand
    'stores da as a SQL dataadapter
    Dim da As New SqlDataAdapter
    'stores dt as a datatable
    Dim dt As New DataTable
    'stores ds as a dataset
    Dim ds As New DataSet
    'connects to the SQL database by calling the DatabaseConnect from the module
    Call DatabaseConnect()

    'sets the SQL statement
    sql = "Select * from Teacher where TeacherID = " & TeacherID & ""
    'sets the connection path as the connectionstring of con from the module
    Cmd.Connection = con
    'sets the command statement as the SQL statement
    Cmd.CommandText = sql
    'the dataadapter then uses this SQL command to Query the database
    da.SelectCommand = Cmd
    'the dataadapter retrieves the query results from the SQL statement after
    connecting to the Database and fills a dataset
    da.Fill(ds, "Teacher")
    'the datatable variable is then set to the table result of the dataset variable
    dt = ds.Tables("Teacher")

    'if the number of rows in the datatable is one or more than the IDexists state is
    set to true otherwise it is set to false
    If dt.Rows.Count >= 1 Then
        IDexists = True
    Else
        IDexists = False
    End If

End Function
'this function returns a true or false state depending on whether the row exists in
the database, and works in a similar manner as the function seen above
Public Function rowexists() As Boolean
    Dim Cmd As New SqlCommand

```

```

Dim da As New SqlDataAdapter
Dim dt As New DataTable
Dim ds As New DataSet

Call DatabaseConnect()
sql = "SELECT DISTINCT Teacher.Firstname, Teacher.Surname FROM Teacher WHERE
Teacher.Firstname = '" & Firstname.Text & "' AND Teacher.Surname = '" & Surname.Text &
"';"
Dim cmmnd As New SqlCommand(sql, con)
da.SelectCommand = cmmnd
da.Fill(ds, "temp")
dt = ds.Tables("temp")
If dt.Rows.Count >= 1 Then
    rowexists = True
Else
    rowexists = False
End If
End Function
'this sub runs when the add_edit button is clicked by the user
Private Sub Edit_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Add_Edit.Click

    Dim da As New SqlDataAdapter
    Dim dt As New DataTable
    Dim ds As New DataSet
    'stores a teacherID that will be used as the teacherID to be inserted in the
    database
    Dim tID As Integer
    'sets the SQL satatement
    sql = "Select TOP 1 TeacherID from Teacher ORDER BY TeacherID DESC"
    'stores cmmnd as a SQL command that uses the above SQL statement and the connection
    string of con
    Dim cmmnd As New SqlCommand(sql, con)
    'sets the dataadapter select command as cmmnd
    da.SelectCommand = cmmnd
    'the dataadapter retrieves the query results from the SQL statement after
    connecting to the Database and fills a dataset
    da.Fill(ds, "Teacher")
    'the datatable variable is then set to the table result of the dataset variable
    dt = ds.Tables("Teacher")
    'The TeacherID is then set as the value of the highest TeacherID found from the
    Query and then incremented by 1, the SQL statement takes into account deleting teachers so
    that inserting a teacher will use the first available TeacherID
    tID = ds.Tables(0).Rows(0).Item("TeacherID").ToString() + 1

    DatabaseConnect()
    'if the textboxes have text and have been validated the program continues
    If txtvalidate() = True Then
        'if the teacherID does not already exist then the program continues with an
        insert statement
        If IDexists(TeacherID) = False Then
            'sets the SQL statement
            sql = "INSERT INTO Teacher(TeacherID, Firstname, Surname) VALUES
(@TeacherID, @Fname, @Sname)"
            'uses cmmnd as a new SQL command that uses the above SQL statement and the
            connection string of con
            Using cmd = New SqlCommand(sql, con)
                'sets the parameters of the vlues to be inserted
                cmd.Parameters.AddWithValue("Fname", Firstname.Text)
                cmd.Parameters.AddWithValue("Sname", Surname.Text)
                cmd.Parameters.AddWithValue("TeacherID", tID)
            End Using
        End If
    End If
End Sub

```

```

        'executes the SQL command
        cmd.ExecuteNonQuery()
    End Using
    'closes the database connection
    con.Close()
    'notifies the user with the appropriate message:
    MsgBox("Record Added to Database", MsgBoxStyle.Information)
    'closes this form and shows the teacher view form
    Teacher.Show()
    Me.Close()
    'but if the teacherID does exist then the program continues with an update
statement
Else
    'sets the SQL statement
    sql = "UPDATE Teacher SET Firstname = @Fname, Surname = @Sname WHERE
TeacherID = " & TeacherID & ";"
    'uses cmmnd as a new SQL command that uses the above SQL statement and the
connection string of con
    Using cmd = New SqlCommand(sql, con)
        'sets the parameters of the vlues to be inserted
        cmd.Parameters.AddWithValue("Fname", Firstname.Text)
        cmd.Parameters.AddWithValue("Sname", Surname.Text)
        'executes the SQL command
        cmd.ExecuteNonQuery()
    End Using
    'closes the database connection
    con.Close()
    'notifies the user with the appropriate message:
    MsgBox("Record Updated", MsgBoxStyle.Information)
End If
Else
    'if the textvalidate has been returned as false then the following error
message displays to the user
    MsgBox("One or more textboxes have not been completed",
MsgBoxStyle.Information, MessageBoxButtons.OK)
End If
End Sub
'this sub runs when the delte button is clicked and works in a similar manner to the
code in the above sub
Private Sub Delete_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Delete.Click
    Dim da As New SqlDataAdapter
    Dim dt As New DataTable
    Dim ds As New DataSet

    DatabaseConnect()

    'checks if the record that the user wants to delete exists in the database
    If rowexists() = True Then

        'this block of code sets the TeacherID of the record that the user wants to
        delete based o the values of the firstname and surname textboxes
        sql = "Select * FROM Teacher WHERE Firstname = '" & Firstname.Text & "' AND
Surname = '" & Surname.Text & "' "
        Dim cmmnd As New SqlCommand(sql, con)
        da.SelectCommand = cmmnd
        da.Fill(ds, "Teacher")
        dt = ds.Tables("Teacher")
        TID = ds.Tables(0).Rows(0).Item("TeacherID").ToString()

        'provides a yes/no dialog box aasking the user if they really want to delete
        the record
    End If
End Sub

```

```
Dim result As Integer = MessageBox.Show("Do you really want to delete this
record?", "Delete", MessageBoxButtons.YesNo, MessageBoxIcon.Warning)
    'if the answer is yes then the following code deletes the record from the
database using the SQL statement below
    If result = DialogResult.Yes Then
        sql = "DELETE FROM Teacher WHERE Teacher.TeacherID = '" & TID & "' AND
Firstname = '" & Firstname.Text & "' and Teacher.Surname = '" & Surname.Text & "';"
        Using cmd = New SqlCommand(sql, con)
            cmd.ExecuteNonQuery()
        End Using
        con.Close()
        MsgBox("Record Deleted From Database", MsgBoxStyle.Information)
        Teacher.Show()
        Me.Close()
        'but if the answer is no nothing happens
    ElseIf result = DialogResult.No Then
        End If
    Else
        'if the record that the user wants to delete does not exist in the database,
the following message is displayed
        MsgBox("The Entered Record does not exist", MsgBoxStyle.Information,
MessageBoxButtons.OK)
    End If

End Sub
End Class
```

Teacher Classes

This form allows a user to display classes from within the database. The classes that are displayed can be filtered via a selection of choices which a class is associated with, this helps to make finding classes easier. The filtered selection is then displayed in a data grid. A class within the system can be edited by clicking on the corresponding 'edit class' button on the row of the record that user wants to edit, when clicked a new form is displayed allowing a user to either delete or edit the record. Furthermore a new class can be inserted into the system by completing the entry boxes within this form and clicking the 'create class' button. The form validates the entry within the textboxes so that a record cannot be inserted if there are empty fields. A printed hard copy of this form can also be produced.

Class Group	Subject	Department	Year	Firstname	Surname	Class Details
Class Group 26	Geography	Humanities	2013	Amelia	Lamberts	Edit Class
Class Group 29	Geography	Humanities	2013	Andrew	Thomson	Edit Class
Class Group 30	Geography	Humanities	2013	Audrey	Rutherford	Edit Class
Class Group 47	Leisure and Tourism	Humanities	2013	Cameron	May	Edit Class
Class Group 31	Health And Social care	Humanities	2013	Cameron	Wallace	Edit Class
Class Group 24	Geography	Humanities	2013	Carolyn	Roberts	Edit Class
Class Group 37	History	Humanities	2013	Christian	Lyman	Edit Class
Class Group 33	History	Humanities	2013	Ella	Stewart	Edit Class
Class Group 46	Leisure and Tourism	Humanities	2013	Grace	Nash	Edit Class
Class Group 60	Religious Studies	Humanities	2013	Joanne	Comish	Edit Class
Class Group 35	History	Humanities	2013	Jonathan	Morison	Edit Class
Class Group 25	Geography	Humanities	2013	Jonathan	Quinn	Edit Class
Class Group 27	Geography	Humanities	2013	Julia	Grant	Edit Class
Class Group 38	History	Humanities	2013	Kimberly	Russell	Edit Class
Class Group 61	Religious Studies	Humanities	2013	Rose	Hemmings	Edit Class
Class Group 32	History	Humanities	2013	Ryan	Powell	Edit Class
Class Group 34	History	Humanities	2013	Sam	Churchill	Edit Class

```

Addclassbutton System.Windows.Forms.Button
AddClassgroup System.Windows.Forms.GroupBox
Addsearch System.Windows.Forms.Button
banner System.Windows.Forms.PictureBox
Classesgrid System.Windows.Forms.DataGridView
Classeslink System.Windows.Forms.LinkLabel
Classfield System.Windows.Forms.TextBox
Classgroupfield System.Windows.Forms.TextBox
Classgrouplabel System.Windows.Forms.Label
Departmentfield System.Windows.Forms.ComboBox
Field System.Windows.Forms.ComboBox
fieldlabel System.Windows.Forms.Label
Firstnamefield System.Windows.Forms.TextBox
Firstnamelabel System.Windows.Forms.Label
Formheader System.Windows.Forms.Label
linkarrow System.Windows.Forms.Label
MainMenu System.Windows.Forms.LinkLabel
Namefield System.Windows.Forms.TextBox
printbutton System.Windows.Forms.Button
PrintDialog System.Windows.Forms.PrintDialog
PrintForm Microsoft.VisualBasic.PowerPacks.Printing.PrintForm
SearchStudent System.Windows.Forms.GroupBox
Subjectfield System.Windows.Forms.ComboBox
Subjectfieldada System.Windows.Forms.ComboBox
Subjectlabel System.Windows.Forms.Label
Surnamefield System.Windows.Forms.TextBox
Surnamelabel System.Windows.Forms.Label
Teacher_Classes System.Windows.Forms.Form
wherelabel System.Windows.Forms.Label
Yearfield System.Windows.Forms.ComboBox

```

Parameter Name	Description
teacher_classes	This code within this sub is run when the form initially loads. This form mainly deals with generating the values of the combo-boxes on the form.
editedcheck	This function sets a true or false condition depending on whether the values in any of the entry boxes differ from their original values. If they differ then a false state is returned.
MainMenu_LinkClicked	When this link label is clicked the user is redirected to the Main menu form. If fields on the form have been edited then the user is asked to confirm their navigation choice.
Classes_CellContentClick	When a cell that is part of the Classes grid is clicked then the Edit Class form displays, the ClassID that is on the row of the data grid button that is clicked is also passed to the edit ClassID form.
Field_SelectedIndexChanged	Whenever the text value of the filter field is changed this sub determines which objects are either visible or non-visible.
search_Click	This sub runs when the Search button is clicked and returns a query matching the criteria the user has used to filter their Class selection by, this selection is then displayed in the Class data grid.
txtvalidate	This function sets a true or false condition depending on whether any textboxes on the form are blank. It also sets the background colour of textboxes appropriately to indicate to the user that fields have not been completed.
name exists	This function returns a true or false state depending on whether the teacher the user has entered as the teacher of a new class actually exists within the system. If it does exist then a row should be returned when the system is queried using the values of the first name and surname textboxes.
Addclassbutton_Click	This sub is responsible for adding the new class record into the database when the Add class button is clicked. A new class ID is dynamically generated and this is used along with the user's class entry values to insert a new record into the database.
printbutton_Click	This sub allows the user to print a hard copy of the form when the print icon is clicked, additional print dialog options are then displayed allowing the user to choose printer and number of copies amongst other settings.

```

'Imports the following system code commands so that I can use them appropriately
Imports System.Data.SqlClient
Imports System.Data
Imports System.Configuration
Imports System.Data.DataTable
Public Class Teacher_Classes
    'Stores the teacherID variable within this form so that it can be used within any sub
    Dim TID As Integer
    'This sub is performed when the form initially loads
    Private Sub teacher_classes(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Me.Load
        'Calls the database connect sub from the module
        DatabaseConnect()
        'the datagrid's visibility is set to false
        Classesgrid.Visible = False
        'stores dt as a variable datatable
        Dim dt As New DataTable
        'sets the screen dimensions of the form to automatically generate center screen of
        'the users display
        Me.Left = (Screen.PrimaryScreen.WorkingArea.Width - Me.Width) / 2
        Me.Top = (Screen.PrimaryScreen.WorkingArea.Height - Me.Height) / 2

        'Uses the Generate combo function of the module to populate a combobox by passing
        'the table, field and needempty variables to the Generatecombo function and then returning
        'the result in a datatable
        dt = GenerateCombo("DatasetID", "DatasetID.DatasetID,Year", True)
        'sets the datasource of the combobox
        Yearfieldadd.DataSource = dt
        'sets the display member of the combobox
        Yearfieldadd.DisplayMember = "Year"
        'sets the valuemember of the combobox, so that the selected index of each combobox
        'row is the chronological ID of the valuemember column ID
        Yearfieldadd.ValueMember = "DatasetID.DatasetID"

        dt = GenerateCombo("Subject", "SubjectID, Subject", True)
        Subjectfieldadd.DataSource = dt
        Subjectfieldadd.DisplayMember = "Subject"
        Subjectfieldadd.ValueMember = "SubjectID"
    End Sub
    'this function returns a true or false state depending on whether the conditions
    'within the sub have been met or not met
    Public Function editedcheck() As Boolean
        Call DatabaseConnect()
        'initially edited check is set to true
        editedcheck = True
        'However if any of the following textbox fields are not blank then the editedcheck
        'variable is set to false
        If Firstnamefield.Text <> "" Then
            editedcheck = False
        End If
        If Surnamefield.Text <> "" Then
            editedcheck = False
        End If
        If Classgroupfield.Text <> "" Then
            editedcheck = False
        End If
        If Subjectfield.Text <> "" Then
            editedcheck = False
        End If
        If Yearfield.Text <> "" Then
            editedcheck = False
        End If
    End Function

```

```

End Function
Private Sub MainMenu_LinkClicked(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles MainMenu.LinkClicked
    'calls the edited check function
    If editedcheck() Then
        'if edited check is true and nothing has been altered on the form then the
following forms either display or close
        Main_Menu.Show()
        Edit_Classes.Close()
        Me.Close()
    ElseIf editedcheck() = False Then
        'however if the editedcheck state is false then a mesagebox with yes and no
options is displayed to the user asking them if they really want to leave the form since
they have written information in the forms textboxes
        Dim result As Integer = MessageBox.Show("Do you really want to change
window?", "Edited Record", MessageBoxButtons.YesNo, MessageBoxIcon.Warning)
        'if the user clicks yes then the following forms are either displayed or
closed
        If result = DialogResult.Yes Then
            Main_Menu.Show()
            Edit_Classes.Close()
            Me.Close()
        'but if the user clicks no then nothing happens and the user reamins on
the current form
        ElseIf result = DialogResult.No Then
            End If
        End If
    End Sub
    'when a button that is part of the button column is clicked, the code within the
following sub is run
    Private Sub Classes_CellContentClick(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.DataGridViewCellEventArgs) Handles Classesgrid.CellContentClick
        'stores the ClassID
        Dim ClassID As String

        If e.ColumnIndex = 7 Then
            Edit_Classes.Close()
            'sets the value of the ClassID as the value of the cell that is on the same
row as the button that has been clicked
            ClassID = Classesgrid.Rows(e.RowIndex).Cells("ClassID").Value
            'sets the ClassID of the edit Class form as the ClassID of this form
            Edit_Classes.ClassID = ClassID
            'show the editstudentform and closes this form
            Edit_Classes.Show()
        End If
    End Sub
    'when the text within the field combobox is changed then the code within this sub is
run
    Private Sub Field_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Field.SelectedIndexChanged
        'stores dt as a datatable
        Dim dt As DataTable
        'sets the wherelabel as visible to the user
        wherelabel.Visible = True

        'This case statement selects which textbox or combobox is visible when the
corrsponding text value of the field combobox is selected and sets all other textboxes or
comboboxes as not visible when a case is selected by the user's mouse click choice
        Select Case Field.Text
            Case ""
                wherelabel.Visible = False

```

```

Case "Teacher Name"
    Namefield.Visible = True
    Subjectfield.Visible = False
    Departmentfield.Visible = False
    Classfield.Visible = False
    Yearfield.Visible = False
    wherelabel.Text = "Name:"
Case "Class"
    Namefield.Visible = False
    Namefield.Visible = False
    Subjectfield.Visible = False
    Departmentfield.Visible = False
    Classfield.Visible = True
    Yearfield.Visible = False
    wherelabel.Text = "Class:"
Case "Subject"
    'Uses the Generate combo function of the module to populate a combobox by
    passing the table, field and needempty variables to the Generatecombo function and then
    returning the result in a datatable
    dt = GenerateCombo("Subject", "SubjectID, Subject", True)
    'sets the datasource of the combobox
    Subjectfield.DataSource = dt
    'sets the display member of the combobox
    Subjectfield.DisplayMember = "Subject"
    'sets the valuemember of the combobox, so that the selected index of each
    combobox row is the chronological ID of the valuemember column ID
    Subjectfield.ValueMember = "SubjectID"

    Namefield.Visible = False
    Namefield.Visible = False
    Subjectfield.Visible = True
    Departmentfield.Visible = False
    Classfield.Visible = False
    Yearfield.Visible = False
    wherelabel.Text = "Subject:"
Case "Department"
    dt = GenerateCombo("Department", "DepartmentID,Department", True)
    Departmentfield.DataSource = dt
    Departmentfield.DisplayMember = "Department"
    Departmentfield.ValueMember = "Department"

    Namefield.Visible = False
    Namefield.Visible = False
    Subjectfield.Visible = False
    Departmentfield.Visible = True
    Classfield.Visible = False
    Yearfield.Visible = False
    wherelabel.Text = "Department:"
End Select
End Sub
'When the search button is clicked the code within the following sub is run
Private Sub search_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Addsearch.Click
    'clears all columns from the Classesgrid
    Classesgrid.Columns.Clear()
    'stores a dataset variable
    Dim ds As New DataSet

    'sets the SQL statement
    sql = "SELECT DISTINCT Class.ClassID, Class.ClassGroup, Subject.Subject,
    Department.Department, DatasetID.Year, Teacher.Firstname, Teacher.Surname FROM Department
    INNER JOIN Subject ON Department.DepartmentID = Subject.DepartmentID INNER JOIN Class

```

```

INNER JOIN DatasetID ON Class.DatasetID = DatasetID.DatasetID ON Subject.SubjectID =
Class.SubjectID INNER JOIN Teacher INNER JOIN Teaches ON Teacher.TeacherID =
Teaches.TeacherID ON Class.TeacherID = Teacher.TeacherID"
    'This Case statement is used to alter the SQL query depending on which text is
selected from the combobox, when a case is selected the above SQL statement and the
corresponding case 'SQL where statement' are concatenated to form one SQL statement
    Select Case Field.Text
        Case ""
            sql = sql & " WHERE DatasetID.DatasetID = '" & moduleDataset & "' ORDER
BY Teacher.Firstname ASC, Teacher.Surname ASC, Class.ClassID ASC"
        Case "Teacher Name"
            sql = sql & " WHERE (Teacher.Firstname LIKE '%" & Namefield.Text & "%' Or
Teacher.Surname LIKE '%" & Namefield.Text & "%') AND DatasetID.DatasetID = '" &
moduleDataset & "' ORDER BY Teacher.Firstname ASC, Teacher.Surname ASC, Class.ClassID ASC"
        Case "Department"
            sql = sql & " WHERE Department.Department = '" & Departmentfield.Text & "'"
AND DatasetID.DatasetID = '" & moduleDataset & "' ORDER BY Teacher.Firstname ASC,
Teacher.Surname ASC, Class.ClassID ASC"
        Case "Subject"
            sql = sql & " WHERE Subject.Subject = '" & Subjectfield.Text & "' AND
DatasetID.DatasetID = '" & moduleDataset & "' ORDER BY Teacher.Firstname ASC,
Teacher.Surname ASC, Class.ClassID ASC"
        Case "Class"
            sql = sql & " WHERE Class.ClassGroup LIKE '%" & Classfield.Text & "%' AND
DatasetID.DatasetID = '" & moduleDataset & "' ORDER BY Teacher.Firstname ASC,
Teacher.Surname ASC, Class.ClassID ASC"
    End Select

    'the dataset is set after passing the following SQL statement and table into the
GenerateDataset sub of the public module
    ds = generateDataset(sql, "temp")
    'sets the datasource of the datagrid
    Classesgrid.DataSource = ds
    'sets the datagrid as visible
    Classesgrid.Visible = True
    'sets the datamember of the datagrid
    Classesgrid.DataMember = "temp"
    'hides the ClassID column from the user
    Classesgrid.Columns("ClassID").Visible = False

    'creates a new variable that stores information for a button column that can be
added to the datagrid
    Dim Columnview As New DataGridViewButtonColumn
    With Columnview
        'sets the header text of the button column
        .HeaderText = "Class Details"
        'sets the name of the button column
        .Name = "Details"
        'sets the text displayed on the buttons of the button column in the datagrid
        .Text = "Edit Class"
        .UseColumnTextForButtonValue = True
    End With

    'adds the button column to the datagrid
    Classesgrid.Columns.Add(Columnview)
    'auto resizes the column widths of the datagrid
    Classesgrid.AutoResizeColumns()
End Sub
'this function returns a true or false state depending on whether the conditions
within the sub have been met or not met
Public Function txtvalidate() As Boolean
    Call DatabaseConnect()

```

```

'initially txtvalidate check is set to true
txtvalidate = True
'if any of the following text boxes are blank then the background colour of them
changes to a red colour indicating an error otherwise their background colours are set to
white
If Subjectfieldadd.Text = "" Then
    Subjectfieldadd.BackColor = Color.Salmon
    txtvalidate = False
Else
    Subjectfieldadd.BackColor = Color.White
End If
If Firstnamefield.Text = "" Then
    Firstnamefield.BackColor = Color.Salmon
    txtvalidate = False
Else
    Firstnamefield.BackColor = Color.White
End If
If Surnamefield.Text = "" Then
    Surnamefield.BackColor = Color.Salmon
    txtvalidate = False
Else
    Surnamefield.BackColor = Color.White
End If
If Classgroupfield.Text = "" Then
    Classgroupfield.BackColor = Color.Salmon
    txtvalidate = False
Else
    Classgroupfield.BackColor = Color.White
End If
If Yearfieldadd.Text = "" Then
    Yearfieldadd.BackColor = Color.Salmon
    txtvalidate = False
Else
    Yearfieldadd.BackColor = Color.White
End If
End Function
'This function returns either a true or false condition depending on whether the user
has entered a valid teacher's name into the name textboxes
Public Function nameexists() As Boolean
    'stores cmd as a sqlCommand
    Dim Cmd As New SqlCommand
    'stores da as a SQL dataadapter
    Dim da As New SqlDataAdapter
    'stores dt as a datatable
    Dim dt As New DataTable
    'stores ds as a dataset
    Dim ds As New DataSet
    'connects to the SQL database by calling the DatabaseConnect from the module
    Call DatabaseConnect()

    'sets the SQL statemnt
    sql = "Select * FROM Teacher WHERE Firstname = '" & Firstnamefield.Text & "' AND
Surname = '" & Surnamefield.Text & "'"
    'stores cmmnd as a SQL command that uses the above SQL statement and the connection
    string of con
    Dim cmmnd As New SqlCommand(sql, con)
    'sets the dataadapter select command as cmmnd
    da.SelectCommand = cmmnd
    'the dataadapter retrieves the query results from the SQL statement after
    connecting to the Database and fills a dataset
    da.Fill(ds, "Teacher")
    'the datatble variable is then set to the table result of the dataset variable

```

```

dt = ds.Tables("Teacher")

If dt.Rows.Count >= 1 Then
    'if there are more than one row in the datatable then the nameexists condition
    'is set to true and TID variable is set to the teacherID variable of the row returned in
    'the Datatable
    nameexists = True
    TID = ds.Tables(0).Rows(0).Item("TeacherID").ToString()
Else
    'if there is not a row then the condition is set to false
    nameexists = False
End If
End Function
'the following sub is run when the Addclassbutton is clicked
Private Sub Addclassbutton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Addclassbutton.Click
    'stores da as a SQL dataadapter
    Dim da As New SqlDataAdapter
    'stores dt as a datatable
    Dim dt As New DataTable
    'stores ds as a dataset
    Dim ds As New DataSet
    'stores the ClassID
    Dim CID As Integer

    'sets the SQL statement query
    sql = "Select TOP 1 ClassID FROM Class ORDER BY ClassID DESC"
    Dim cmmnd As New SqlCommand(sql, con)
    da.SelectCommand = cmmnd
    da.Fill(ds, "Class")
    dt = ds.Tables("Class")
    'The ClassID is then set as the value of the highest ClassID found from the Query
    'and then incremented by 1, the SQL statement takes into account removing classes so that
    'inserting a record will use the first available ClassID
    CID = ds.Tables(0).Rows(0).Item("ClassID").ToString() + 1

    DatabaseConnect()

    'if both of the following conditions have been met then the insert continues
    If txtvalidate() = True Then
        If nameexists() = True Then

            'sets the SQL insert statement
            sql = "INSERT INTO Class (ClassID, SubjectID, TeacherID, ClassGroup,
DatasetID) VALUES (@ClassID, @SubjectID, @TeacherID, @ClassG, @Year)"
            'uses cmmnd as a new SQL command that uses the above SQL statement and the
            connection string of con
            Using cmd = New SqlCommand(sql, con)
                'sets the parameters of the values to be inserted
                cmd.Parameters.AddWithValue("ClassID", CID)
                'the value of this parameter is set to the selected value member of the
                combobox
                cmd.Parameters.AddWithValue("SubjectID",
Subjectfieldadd.SelectedValue)
                cmd.Parameters.AddWithValue("TeacherID", TID)
                cmd.Parameters.AddWithValue("ClassG", Classgroupfield.Text)
                cmd.Parameters.AddWithValue("Year", Yearfieldadd.SelectedValue)
                'execute the SQL command
                cmd.ExecuteNonQuery()
            End Using
            con.Close()
        End If
    End If
End Sub

```

```
'displays a messagebox to the user to verify that the record has been
added to the database
    MsgBox("Record Added", MsgBoxStyle.Information)
    'this command clicks the search button, which essentially refreshes the
datagrid so that the user can visibly see that their record has been added
    Addsearch.PerformClick()
Else
    'if the namevalidate state is not true then the following messagebox
displays to the user
    MsgBox("The Entered Teacher Name does not exist", MsgBoxStyle.Information,
MessageBoxButtons.OK)
End If
Else
    'if the txtvalidate state is not true then the following messagebox displays
to the user
    MsgBox("One or more textboxes have not been completed",
MsgBoxStyle.Information, MessageBoxButtons.OK)
End If

End Sub

'deals with the print form button
Private Sub printbutton_Click(sender As System.Object, e As System.EventArgs) Handles
printbutton.Click
    'sets the printer settings to default
    PrintForm.PrinterSettings = PrintDialog.PrinterSettings
    If PrintDialog.ShowDialog() = DialogResult.OK Then
        'allows the user to change print settings
        PrintForm.PrinterSettings = PrintDialog.PrinterSettings
        'changes the orientation of the printed document to landscape
        Me.PrintForm.PrinterSettings.DefaultPageSettings.Landscape = True
        'refreshes the form so that the printdialog is not printed along with the form
        Me.Refresh()
        'prints the current form
        Me.PrintForm.Print()
    End If
End Sub
End Class
```

Edit Classes

This form appears when a user wants to edit an existing class within the system. The Class ID of the class that the user wants to edit is passed into this form so that the textbox and combo box values can be set appropriately as the values corresponding to the record of the Class ID. From here the values of any of the entry boxes can be altered but they cannot be left blank. When the Edit class button is clicked the record is updated using the values entered. An existing class can also be deleted on this form by clicking the delete button, but first it will check that if a record matching the entered values actually exists within the database system, if it does then the record will be deleted.

The screenshot shows a Windows application window titled "Edit Classes". The window has a title bar with standard minimize, maximize, and close buttons. Below the title bar is a navigation bar containing "Main Menu", "Teacher Classes", and "Edit Classes". The main area is titled "Edit Classes" in red. It contains five input fields with labels in red:

- Teacher Firstname:
- Teacher Surname:
- Class Group:
- Subject:
- Year:

At the bottom are two blue rounded rectangular buttons labeled "Edit Class" and "Delete".

```

Banner System.Windows.Forms.PictureBox
Classeslink System.Windows.Forms.LinkLabel
Classgroupfield System.Windows.Forms.TextBox
Classgrouplabel System.Windows.Forms.Label
Deletebtn System.Windows.Forms.Button
Edit System.Windows.Forms.Button
Edit_Classes System.Windows.Forms.Form
Editclasseslink System.Windows.Forms.LinkLabel
edittitle System.Windows.Forms.Label
Firstnamefield System.Windows.Forms.TextBox
Firtnamelabel System.Windows.Forms.Label
linkarrow System.Windows.Forms.Label
linkarrow2 System.Windows.Forms.Label
MainMenu System.Windows.Forms.LinkLabel
Subjectfield System.Windows.Forms.ComboBox
Subjectlabel System.Windows.Forms.Label
Surnamefield System.Windows.Forms.TextBox
Surnamelabel System.Windows.Forms.Label
Yearfield System.Windows.Forms.ComboBox
Yearlabel System.Windows.Forms.Label
Centre number: 23210

```

Parameter Name	Description
_ClassID	This sub uses a get command to fetch the corresponding Class ID of the row the user wants to edit and stores it as a private integer, which will be used to reference the correct row in the database.
Classeslink_LinkClicked	When this link label is clicked the user is redirected to the Teacher Classes form. If fields on the form have been edited then the user is asked to confirm their navigation choice.
MainMenu_LinkClicked	When this link label is clicked the user is redirected to the Main menu form. If fields on the form have been edited then the user is asked to confirm their navigation choice.
Edit_classes_Load	This code encased within this sub initiates when the form loads. A dataset variable is filled with the corresponding Class edit values of the Class ID that is passed to the form. Also if there is a Class ID passed to the form then the textboxes display the values of Class, subject etc., of the Class the user wants to edit. This load event also handles the generating of the combo boxes and the items within them.
editedcheck	This function sets a true or false condition depending on whether the values in any of the entry boxes differ from their original values. If they differ then a false state is returned.
name exists	This function returns a true or false state depending on whether the teacher the user has entered as the teacher of a new class actually exists within the system. If it does exist then a row should be returned when the system is queried using the values of the first name and surname textboxes.
txtvalidate	This function sets a true or false condition depending on whether any textboxes on the form are blank. It also sets the background colour of textboxes appropriately to indicate to the user that fields have not been completed.
rowexists	This function checks the database to see if the record that the user wants to delete actually exists within the database.
Edit_Click	When the edit class button is clicked the record that the user has selected in the database is updated with the values they have entered in the forms entry boxes.
Deletebtn_Click	When the delete button is clicked by the user, and the row exists, then a SQL statement removes the record from the database.

```

'Imports the following system code commands so that I can use them appropriately
Imports System.Data.SqlClient
Imports System.Data
Imports System.Configuration
Imports System.Data.DataTable
Public Class Edit_Classes
    'sets the variable that is received from the Grade form as a private variable which
    can be used in any sub within this class
    Private _ClassID As Integer
    'stores two separate TeacherID variables within this class
    Dim TeacherID As Integer
    Dim TID As Integer

    'sets the values of several variables that can be used to check the values of these
    variables later in the code, they can be used in any sub
    Dim Fnamecheck As String
    Dim Snamecheck As String
    Dim Classgroupcheck As String
    Dim Subjectcheck As String
    Dim Yearcheck As String
    'creates a property of the variable that is being received
    Public Property ClassID() As Integer
        Get
            'gets the variable from the form that has been specified, in this case the
            grade form
            Return _ClassID
        End Get
        Set(ByVal value As Integer)
            'sets the variable that has been received by the get statement as the
            equivalent variable that will be used in this form
            _ClassID = value
        End Set
    End Property
    'this sub runs when the Teacher classes link is clicked
    Private Sub Classeslink_LinkClicked(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles Classeslink.LinkClicked
        'the following forms either display or close if edited check is true
        If editedcheck() Then
            Teacher_Classes.Show()
            Me.Close()
        ElseIf editedcheck() = False Then
            'if edited check is not true then a dialog message displays to the user with a
            yes/no option
            Dim result As Integer = MessageBox.Show("Do you really want to change
window?", "Edited Record", MessageBoxButtons.YesNo, MessageBoxIcon.Warning)
            'if the user clicks yes then the following forms either display or close
            If result = DialogResult.Yes Then
                Teacher_Classes.Show()
                Me.Close()
                'if the user clicks no then nothing happens
            ElseIf result = DialogResult.No Then
                End If
            End If
        End Sub
        'this sub runs when the main menu link is clicked and functions in the same manner as
        the teacher link above
        Private Sub MainMenu_LinkClicked(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles MainMenu.LinkClicked
            If editedcheck() Then
                Teacher_Classes.Close()
                Me.Close()
                Main_Menu.Show()
            End If
        End Sub
    End Class

```

```

        ElseIf editedcheck() = False Then
            Dim result As Integer = MessageBox.Show("Do you really want to change
window?", "Edited Record", MessageBoxButtons.YesNo, MessageBoxIcon.Warning)
            If result = DialogResult.Yes Then
                Teacher_Classes.Close()
                Me.Close()
                Main_Menu.Show()
            ElseIf result = DialogResult.No Then
                End If
            End If
        End Sub
        'This sub runs when the Edit classes form is initially loaded
        Private Sub Edit_classes_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Me.Load

            'sets the screen dimensions of the form to automatically generate center screen of
the users display
            Me.Left = (Screen.PrimaryScreen.WorkingArea.Width - Me.Width) / 2
            Me.Top = (Screen.PrimaryScreen.WorkingArea.Height - Me.Height) / 2
            'calls database connect from the public module
            DatabaseConnect()
            'stores dt as a datatable
            Dim dt As DataTable
            'stores ds as a dataset
            Dim ds As DataSet

            'Uses the Generate combo function of the module to populate a combobox by passing
the table, field and needempty variables to the Generatecombo function and then returning
the result in a datatable
            dt = GenerateCombo("DatasetID", "DatasetID.DatasetID,Year", True)
            'sets the datasource of the combobox
            Yearfield.DataSource = dt
            'sets the display member of the combobox
            Yearfield.DisplayMember = "Year"
            'sets the index value of each item in the Gradetype combobox to match their
corresponding ID's
            Yearfield.ValueMember = "DatasetID.DatasetID"

            dt = GenerateCombo("Subject", "SubjectID, Subject", True)
            Subjectfield.DataSource = dt
            Subjectfield.DisplayMember = "Subject"
            Subjectfield.ValueMember = "SubjectID"

            ds = Editclassvalues(ClassID)
            Fnamecheck = ds.Tables(0).Rows(0).Item("Firstname").ToString()
            Snamecheck = ds.Tables(0).Rows(0).Item("Surname").ToString()
            Classgroupcheck = ds.Tables(0).Rows(0).Item("ClassGroup").ToString()
            Subjectcheck = ds.Tables(0).Rows(0).Item("Subject").ToString()
            Yearcheck = ds.Tables(0).Rows(0).Item("Year").ToString()

            Firstnamefield.Text = Fnamecheck
            Surnamefield.Text = Snamecheck
            Subjectfield.Text = Subjectcheck
            Classgroupfield.Text = Classgroupcheck
            Yearfield.Text = Yearcheck

        End Sub
        'this function returns a true or false state depending on whether conditions within
the function have been met or not
        Public Function editedcheck() As Boolean
            Call DatabaseConnect()
            'initially the functions state is set to true
    
```

```
editedcheck = True
'but if any of the following textbox fields values are not the same as their
respective stored variables then the functions state is set to false
If Firstnamefield.Text <> Fnamecheck Then
    editedcheck = False
End If
If Surnamefield.Text <> Snamecheck Then
    editedcheck = False
End If
If Subjectfield.Text <> Subjectcheck Then
    editedcheck = False
End If
If Classgroupfield.Text <> Classgroupcheck Then
    editedcheck = False
End If
If Yearfield.Text <> Yearcheck Then
    editedcheck = False
End If
End Function
'this function returns a true or false state depending on whether conditions within
the function have been met or not
Private Function txtvalidate() As Boolean
    Call DatabaseConnect()
    'initially the functions state is set to true
    txtvalidate = True

    'if any of the textboxes are left blank then the function returns a false state
    and the background of the textbox field changes to red, otherwise the background of the
    textbox changes to white
    If Firstnamefield.Text = "" Then
        Firstnamefield.BackColor = Color.Salmon
        txtvalidate = False
    Else
        Firstnamefield.BackColor = Color.White
    End If
    If Surnamefield.Text = "" Then
        Surnamefield.BackColor = Color.Salmon
        txtvalidate = False
    Else
        Surnamefield.BackColor = Color.White
    End If
    If Classgroupfield.Text = "" Then
        Classgroupfield.BackColor = Color.Salmon
        txtvalidate = False
    Else
        Classgroupfield.BackColor = Color.White
    End If
    If Subjectfield.Text = "" Then
        Subjectfield.BackColor = Color.Salmon
        txtvalidate = False
    Else
        Subjectfield.BackColor = Color.White
    End If
    If Yearfield.Text = "" Then
        Yearfield.BackColor = Color.Salmon
        txtvalidate = False
    Else
        Yearfield.BackColor = Color.White
    End If
End Function
```

```

'this function returns a true or false state depending on whether the Teacher name
exists
Public Function nameexists() As Boolean
    'stores cmd as a sqlCommand()
    Dim Cmd As New SqlCommand
    'stores da as a SQL dataadapter
    Dim da As New SqlDataAdapter
    'stores dt as a datatable
    Dim dt As New DataTable
    'stores ds as a dataset
    Dim ds As New DataSet
    'connects to the SQL database by calling the DatabaseConnect from the module
    Call DatabaseConnect()

    'sets the SQL statemnt
    sql = "Select * FROM Teacher WHERE Firstname = '" & Firstnamefield.Text & "' AND
Surname = '" & Surnamefield.Text & "'"
    'stores a SQL command that uses the SQL statement to query the database that is
found when following the connection string of con
    Dim cmmnd As New SqlCommand(sql, con)
    'the dataadapter then uses this SQL command to Query the database
    da.SelectCommand = cmmnd
    'the dataadapter retrieves the query results from the SQL statement after
connecting to the Database and fills a dataset
    da.Fill(ds, "Teacher")
    'the datatable variable is then set to the table result of the dataset variable
    dt = ds.Tables("Teacher")
    'if the nubmer of rows in the datatable is one or more then the IDexists state is
set to true otherwise it is set to false
    If dt.Rows.Count >= 1 Then
        nameexists = True
        TID = ds.Tables(0).Rows(0).Item("TeacherID").ToString()
    Else
        nameexists = False
    End If
End Function
'this function returns a true or false state depending on whether the row exists in
the database, and works in a similar mannar as the function seen above
Public Function rowexists() As Boolean
    Dim Cmd As New SqlCommand
    Dim da As New SqlDataAdapter
    Dim dt As New DataTable

    Dim ds As New DataSet
    Call DatabaseConnect()
    sql = "SELECT DISTINCT Class.ClassID, Teacher.Firstname, Teacher.Surname,
Class.ClassGroup, Subject.Subject, Department.Department, DatasetID.Year FROM Department
INNER JOIN Subject ON Department.DepartmentID = Subject.DepartmentID INNER JOIN Class
INNER JOIN DatasetID ON Class.DatasetID = DatasetID.DatasetID ON Subject.SubjectID =
Class.SubjectID INNER JOIN Teacher INNER JOIN Teaches ON Teacher.TeacherID =
Teaches.TeacherID ON Class.TeacherID = Teacher.TeacherID WHERE Teacher.Firstname = '" &
Firstnamefield.Text & "' AND Teacher.Surname = '" & Surnamefield.Text & "' AND
Class.SubjectID = '" & Subjectfield.SelectedValue & "' AND Class.DatasetID = '" &
Yearfield.SelectedValue & "' AND Class.ClassGroup = '" & Classgroupfield.Text & "'"
    Dim cmmnd As New SqlCommand(sql, con)
    da.SelectCommand = cmmnd
    da.Fill(ds, "temp")
    dt = ds.Tables("temp")
    If dt.Rows.Count >= 1 Then
        rowexists = True
    Else
        rowexists = False
End Function

```

```

        End If
    End Function
    'this sub runs when the edit button is clicked by the user
    Private Sub Edit_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Edit.Click
    Dim da As New SqlDataAdapter
    Dim dt As New DataTable
    Dim ds As New DataSet
    DatabaseConnect()

    'if the textboxes on the form are all not blank then the code continues
    If txtvalidate() = True Then
        ''If the TeacherName exists then the code continues
        If nameexists() = True Then
            'sets the SQL statement
            sql = "UPDATE Class SET Class.TeacherID = @TeacherID, Class.ClassGroup =
@ClassG, Class.SubjectID = @GradeT, Class.DatasetID = @Year WHERE Class.ClassID = '' &
ClassID & ''
            'uses cmd as a new SQL command that uses the above SQL statement and the
connection string of con
            Using cmd = New SqlCommand(sql, con)
                'sets the parameters of the vlues to be inserted
                cmd.Parameters.AddWithValue("TeacherID", TID)
                cmd.Parameters.AddWithValue("Year", Yearfield.SelectedValue)
                cmd.Parameters.AddWithValue("Subject", Subjectfield.SelectedValue)
                cmd.Parameters.AddWithValue("ClassG", Classgroupfield.Text)
                'executes the SQL command
                cmd.ExecuteNonQuery()
            End Using
            'closes connection with the database
            con.Close()
            'informs the user with a message that the record has been updated
            MsgBox("Record Updated", MsgBoxStyle.Information)
        Else
            'but if the teacher name entered by the user does not exist then the
following error message displays and nothing happens
            MsgBox("The Entered Teacher Name does not exist", MsgBoxStyle.Information,
MessageBoxButtons.OK)
        End If
    Else
        'if any of the textbox fields are left blank then the following error message
displays and nothing happens
        MsgBox("One or more textboxes have not been completed",
MsgBoxStyle.Information, MessageBoxButtons.OK)
    End If

End Sub
    'this sub runs when the delete button is clicked and works in a similar manner to the
code in the above sub
    Private Sub Deletebtn_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Deletebtn.Click
    Dim da As New SqlDataAdapter
    Dim dt As New DataTable
    Dim ds As New DataSet
    DatabaseConnect()

    'checks to see if the record actually exists based on the information in the
textboxes
    If rowexists() = True Then
        'this block of code sets the StudentID of the record that the user wants to
delete based on the values of the firstname and surname textboxes

```

```
sql = "Select * FROM Teacher WHERE Firstname = '" & Firstnamefield.Text & "'  
AND Surname = '" & Surnamefield.Text & "' "  
Dim cmmnd As New SqlCommand(sql, con)  
da.SelectCommand = cmmnd  
da.Fill(ds, "Teacher")  
dt = ds.Tables("Teacher")  
TID = ds.Tables(0).Rows(0).Item("TeacherID").ToString()  
  
'provides a yes/no dialog box asking the user if they really want to delete  
the record  
Dim result As Integer = MessageBox.Show("Do you really want to delete this  
record?", "Delete", MessageBoxButtons.YesNo, MessageBoxIcon.Warning)  
'if the answer is yes then the following code deletes the record from the  
database using the SQL statement below  
If result = DialogResult.Yes Then  
    sql = "DELETE FROM Class WHERE TeacherID = '" & TID & "' AND SubjectID =  
    "' & Subjectfield.SelectedValue & "' AND DatasetID = '" & Yearfield.SelectedValue & "' AND  
    ClassGroup = '" & Classgroupfield.Text & "'"  
    Using cmd = New SqlCommand(sql, con)  
        cmd.ExecuteNonQuery()  
    End Using  
    con.Close()  
    MsgBox("Record Deleted From Database", MsgBoxStyle.Information)  
    'the following forms are either displayed or closed  
    Teacher_Classes.Show()  
    Me.Close()  
    'but if the answer is no nothing happens  
ElseIf result = DialogResult.No Then  
    End If  
Else  
    'if the record does not exist then the following error message displays and  
    nothing happens  
    MsgBox("The Entered Record does not exist", MsgBoxStyle.Information,  
    MessageBoxButtons.OK)  
    End If  
End Sub  
End Class
```

Student Classes

The student classes' form allows a user of the system to either view all of a student's classes, in which case they must first search and select a student, or add and remove students from classes. To view a student's classes the user can use a series of filter options which will filter the data grid selection. After a selection of student's has been displayed a user can then view all the classes that a student is a part of. A user can also add or remove a student from a class. The class selection can be filtered, and a user can display a selection of classes. When a class has been found a user can either view the students within that class or add a student to that class. If they decide to view all the students within that class they can then choose to remove a student from the class, in which case a new form will appear. If the user wishes to add a student to the class they first must search a class and select the class they want to add a student to, in which case the form which deals with adding students to a class will show. This form can also be printed so that a hard copy of search criteria can be seen.

Firstname	Surname	Gender	DateOfBirth	Form	Year	
William	ABERCROMBIE	M	31/01/1997	13R	13	<input type="button" value="View Classes"/>
William	AKRAM	M	11/02/1998	13M	13	<input type="button" value="View Classes"/>
William	ALLISON	M	24/05/1998	13B	13	<input type="button" value="View Classes"/>
Will	ALTOFT	M	13/09/1997	13C	13	<input type="button" value="View Classes"/>
Tristan	ASHWORTH	M	23/06/1998	13D	13	<input type="button" value="View Classes"/>
Tom	ATTWOOD	M	13/03/1998	13J	13	<input type="button" value="View Classes"/>
Toby	BAIN	M	11/09/1997	13W	13	<input type="button" value="View Classes"/>
Timothy	BALL-WOOD	M	22/12/1997	13L	13	<input type="button" value="View Classes"/>
Thomas	BARBOUR	M	03/03/1998	13M	13	<input type="button" value="View Classes"/>
Thomas	BARKE	M	18/05/1998	13R	13	<input type="button" value="View Classes"/>
Summer	BELL	F	29/12/1996	13L	13	<input type="button" value="View Classes"/>
Sophie	BIRKS	F	05/05/1998	13W	13	<input type="button" value="View Classes"/>
Sophie	BOND	F	07/09/1997	13A	13	<input type="button" value="View Classes"/>
Simon	BRAY	M	02/07/1998	13D	13	<input type="button" value="View Classes"/>
Samuel	BRENTNALL	M	20/02/1998	13J	13	<input type="button" value="View Classes"/>
Sam	BRODERICK	M	10/09/1997	13D	13	<input type="button" value="View Classes"/>
Sam	BRODERICK	M	29/03/1998	13W	13	<input type="button" value="View Classes"/>

```

addremovefield System.Windows.Forms.ComboBox
banner System.Windows.Forms.PictureBox
Classesgrid System.Windows.Forms.DataGridView
Classeslink System.Windows.Forms.LinkLabel
Classfield System.Windows.Forms.TextBox
Departmentfield System.Windows.Forms.ComboBox
fieldlabel System.Windows.Forms.Label
fieldlabel1 System.Windows.Forms.Label
Formheader System.Windows.Forms.Label
instructlabel System.Windows.Forms.Label
linkarrow System.Windows.Forms.Label
MainMenu System.Windows.Forms.LinkLabel
Namefield System.Windows.Forms.TextBox
printbutton System.Windows.Forms.Button
PrintDialog System.Windows.Forms.PrintDialog
PrintForm Microsoft.VisualBasic.PowerPacks.Printing.PrintForm
Search System.Windows.Forms.Button
SearchForm System.Windows.Forms.ComboBox
SearchGender System.Windows.Forms.ComboBox
SearchName System.Windows.Forms.TextBox
SearchStudent System.Windows.Forms.GroupBox
Searchstudentclasses System.Windows.Forms.Button
SearchYear System.Windows.Forms.ComboBox
Student_Classes System.Windows.Forms.Form
studentclassesfield System.Windows.Forms.ComboBox
Studentclassesgrid System.Windows.Forms.DataGridView
Subjectfield System.Windows.Forms.ComboBox
Viewstudentsgrid System.Windows.Forms.DataGridView
viewstudentsgroup System.Windows.Forms.GroupBox
wherelabel System.Windows.Forms.Label

```

Parameter Name	Description
MainMenu_LinkClicked	When the main menu link label is clicked all open forms close and the main menu displays.
Student_classes	When the form initially loads the screen dimensions are set so that they are centre screen and SQL database is connected to.
Field_SelectedIndexChanged	Whenever the text value of the filter field is changed this sub determines which objects are either visible or non-visible.
search_Click	This sub runs when the Search button is clicked and returns a query matching the criteria the user has used to filter their Student selection by, this selection is then displayed in the data grid.
delete_CellContentClick	If the user wishes to remove a student from the selected class then this instance handles when the button for removing a student is clicked. It passes the students Class ID, name and form to the remove student class form and displays the remove student class form.
Classes_CellContentClick	This sub handles the instance a user clicks one of the buttons on the student class's grid, if the user wants to view students within a class then the data grid updates. If a user wants to add a student to a selected class then the add student class form displays and the Class ID of the class that the user wants to add a student to is passed to the Add student class form.
studentclassesfield_SelectedIndexChanged	Whenever the text value of the filter field is changed this sub determines which objects are either visible or non-visible.
Searchstudentclasses_Click	This sub runs when the Search button is clicked and returns a query matching the criteria the user has used to filter their Class selection by, this selection is then displayed in the data grid.
studentsClasses_CellContentClick	When a button that is part of the student classes data grid is clicked this form displays the selected student's classes and shows them in the data grid.
printbutton_Click	This sub allows the user to print a hard copy of the form when the print icon is clicked, additional print dialog options are then displayed allowing the user to choose printer and number of copies amongst other settings.

```

Imports System.Data.SqlClient
Imports System.Data
Public Class Student_Classes
    Private Sub MainMenuItem_Clicked(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles MainMenuItem.Clicked
        'when the main menu link is clicked the following forms are either displayed or
        closed
        Main_Menu.Show()
        Remove_Student_Class.Close()
        Add_Student_Class.Close()
        Me.Close()
    End Sub
    'This sub is performed when the form initially loads
    Private Sub Student_classes(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Me.Load
        DatabaseConnect()
        'the datagrid's visibility is set to false
        Classesgrid.Visible = False
        'sets the screen dimensions of the form to automatically generate center screen of
        the users display
        Me.Left = (Screen.PrimaryScreen.WorkingArea.Width - Me.Width) / 2
        Me.Top = (Screen.PrimaryScreen.WorkingArea.Height - Me.Height) / 2
    End Sub
    'when the text within the field combobox is changed then the code within this sub is
    run
    Private Sub Field_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles addremoveField.SelectedIndexChanged
        'stores dt as a datatable
        Dim dt As DataTable
        wherelabel.Visible = True

        'This case statement selects which textbox or combobox is visible when the
        corresponding text value of the field combobox is selected and sets all other textboxes or
        comboboxes as not visible when a case is selected by the user's mouse click choice
        Select Case addremoveField.Text
            Case ""
                wherelabel.Visible = False

            Case "Teacher Name"
                Namefield.Visible = True
                Subjectfield.Visible = False
                Departmentfield.Visible = False
                Classfield.Visible = False
                Yearfield.Visible = False
                wherelabel.Text = "Name:"

            Case "Class"
                Namefield.Visible = False
                Namefield.Visible = False
                Subjectfield.Visible = False
                Departmentfield.Visible = False
                Classfield.Visible = True
                Yearfield.Visible = False
                wherelabel.Text = "Class:"

            Case "Subject"
                'Uses the Generate combo function of the module to populate a combobox by
                passing the table, field and needempty variables to the Generatecombo function and then
                returning the result in a datatable
                dt = GenerateCombo("Subject", "SubjectID, Subject", True)
                'sets the datasource of the combobox
                Subjectfield.DataSource = dt
                'sets the display member of the combobox
                Subjectfield.DisplayMember = "Subject"
        End Select
    End Sub

```

```

'sets the valuemember of the combobox, so that the selected index of each
combobox row is the chronological ID of the valuemember column ID
Subjectfield.ValueMember = "SubjectID"

Namefield.Visible = False
Namefield.Visible = False
Subjectfield.Visible = True
Departmentfield.Visible = False
Classfield.Visible = False
Yearfield.Visible = False
wherelabel.Text = "Subject:"

Case "Department"
dt = GenerateCombo("Department", "DepartmentID,Department", True)
Departmentfield.DataSource = dt
Departmentfield.DisplayMember = "Department"
Departmentfield.ValueMember = "Department"

Namefield.Visible = False
Namefield.Visible = False
Subjectfield.Visible = False
Departmentfield.Visible = True
Classfield.Visible = False
Yearfield.Visible = False
wherelabel.Text = "Department:"

Case "Year"
dt = GenerateCombo("DatasetID", "DatasetID.DatasetID,Year", True)
Yearfield.DataSource = dt
Yearfield.DisplayMember = "Year"
Yearfield.ValueMember = "DatasetID.DatasetID"

Namefield.Visible = False
Namefield.Visible = False
Subjectfield.Visible = False
Departmentfield.Visible = False
Classfield.Visible = False
Yearfield.Visible = True
wherelabel.Text = "Year:"

End Select
End Sub
'When the search button is clicked the code within the following sub is run
Private Sub search_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Search.Click
'clears all columns from the Classesgrid
Classesgrid.Columns.Clear()
'stores a dataset variable
Dim ds As New DataSet
'sets the SQL statement
sql = "SELECT DISTINCT Class.ClassID, Class.ClassGroup, Subject.Subject,
Department.Department, DatasetID.Year, Teacher.Firstname, Teacher.Surname FROM Department
INNER JOIN Subject ON Department.DepartmentID = Subject.DepartmentID INNER JOIN Class
INNER JOIN DatasetID ON Class.DatasetID = DatasetID.DatasetID ON Subject.SubjectID =
Class.SubjectID INNER JOIN Teacher INNER JOIN Teaches ON Teacher.TeacherID =
Teaches.TeacherID ON Class.TeacherID = Teacher.TeacherID"
'This Case statement is used to alter the SQL query depending on which text is
selected from the combobox, when a case is selected the above SQL statement and the
corresponding case 'SQL where statement' are concatenated to form one SQL statement
Select Case addremoveField.Text
Case ""
    sql = sql & " WHERE DatasetID.DatasetID = '" & moduleDataset & "' ORDER
BY Teacher.Firstname ASC, Teacher.Surname ASC, Class.ClassID ASC"
Case "Teacher Name"

```

```

        sql = sql & " WHERE Teacher.Firstname LIKE '%" & Namefield.Text & "%' Or
Teacher.Surname LIKE '%' & Namefield.Text & "%' and DatasetID.DatasetID = '' &
moduleDataset & '' ORDER BY Teacher.Firstname ASC, Teacher.Surname ASC, Class.ClassID
ASC"
        Case "Year"
            sql = sql & " WHERE DatasetID.Year = '" & Yearfield.Text & "' and
DatasetID.DatasetID = '' & moduleDataset & '' ORDER BY Teacher.Firstname ASC,
Teacher.Surname ASC, Class.ClassID ASC"
        Case "Department"
            sql = sql & " WHERE Department.Department = '" & Departmentfield.Text & "' and
DatasetID.DatasetID = '' & moduleDataset & '' ORDER BY Teacher.Firstname ASC,
Teacher.Surname ASC, Class.ClassID ASC"
        Case "Subject"
            sql = sql & " WHERE Subject.Subject = '" & Subjectfield.Text & "' and
DatasetID.DatasetID = '' & moduleDataset & '' ORDER BY Teacher.Firstname ASC,
Teacher.Surname ASC, Class.ClassID ASC"
        Case "Class"
            sql = sql & " WHERE Class.ClassGroup LIKE '%' & Classfield.Text & "%' and
DatasetID.DatasetID = '' & moduleDataset & '' ORDER BY Teacher.Firstname ASC,
Teacher.Surname ASC, Class.ClassID ASC"
    End Select

    'the dataset is set after passing the following SQL statement and table into the
GenerateDataset sub of the public module
    ds = generateDataset(sql, "temp")
    'sets the datasource of the datagrid
    Classesgrid.DataSource = ds
    'sets the datagrid as visible
    Classesgrid.Visible = True
    'sets the other datagrids as not visible
    Studentclassesgrid.Visible = False
    Viewstudentsgrid.Visible = False
    'sets the datamember of the datagrid
    Classesgrid.DataMember = "temp"
    'hides the ClassID column from the user
    Classesgrid.Columns("ClassID").Visible = False

    'creates a new variable that stores information for a button column that can be
added to the datagrid
    Dim StudentColumn As New DataGridViewButtonColumn
    With StudentColumn
        'sets the header text of the button column
        .HeaderText = "View Current Students"
        'sets the name of the button column
        .Name = "Details"
        'sets the text displayed on the buttons of the button column in the datagrid
        .Text = "View Students"
        .UseColumnTextForButtonValue = True
    End With

    'adds the button column to the datagrid
    Classesgrid.Columns.Add(StudentColumn)

    Dim Columnview As New DataGridViewButtonColumn
    With Columnview
        .HeaderText = "Add a Student to this Class"
        .Name = "Details"
        .Text = "Add Student"
        .UseColumnTextForButtonValue = True
    End With

    Classesgrid.Columns.Add(Columnview)

```

```

    'auto resizes the column widths of the datagrid
    Classesgrid.AutoResizeColumns()
End Sub
'when the remove student from class button is clicked, the code within the following
sub is run
Private Sub delete_CellContentClick(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.DataGridViewCellEventArgs) Handles Viewstudentsgrid.CellContentClick
    'stores ds as a new dataset
    Dim ds As New DataSet
    'stores the ClassID
    Dim ClassID As String
    'stores the firstname of a student
    Dim fname As String
    'Stores the surname of the student
    Dim sname As String
    'stores the form of the student
    Dim form As String

    If e.ColumnIndex = 8 Then
        Remove_Student_Class.Close()
        'sets the value of these four variables as the value of the cell with the
        selected column name that is on the same row as the button that has been clicked
        ClassID = Viewstudentsgrid.Rows(e.RowIndex).Cells("ClassID").Value
        fname = Viewstudentsgrid.Rows(e.RowIndex).Cells("Firstname").Value
        sname = Viewstudentsgrid.Rows(e.RowIndex).Cells("Surname").Value
        form = Viewstudentsgrid.Rows(e.RowIndex).Cells("form").Value

        'sets the following four variables of the remove student class form as the
        values of their variables found from thedatagrid of this form
        Remove_Student_Class.ClassID = ClassID
        Remove_Student_Class.Fname = fname
        Remove_Student_Class.Sname = sname
        Remove_Student_Class.Form = form
        Remove_Student_Class.Show()
    End If

End Sub
'this sub runs the following code if one of the two buttons on the classes grid is
clicked
Private Sub Classes_CellContentClick(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.DataGridViewCellEventArgs) Handles Classesgrid.CellContentClick
    'stores ds as a dataset
    Dim ds As New DataSet
    'stores the classID
    Dim ClassID As String

    'if the selected index of the button that is clicked is the seventh column then
    the following actionis performed
    If e.ColumnIndex = 7 Then
        ClassID = Classesgrid.Rows(e.RowIndex).Cells("ClassID").Value
        'clears the currentdatagrid column selection
        Viewstudentsgrid.Columns.Clear()
        Classesgrid.Columns.Clear()

        'sets the SQL statement query
        sql = "SELECT Class.ClassID, Student.Firstname, Student.Surname, Student.Form,
Class.ClassGroup, Subject.Subject, Department.Department, DatasetID.Year FROM StudentGroup
INNER JOIN Class ON StudentGroup.ClassID = Class.ClassID INNER JOIN Subject ON
Class.SubjectID = Subject.SubjectID INNER JOIN Department ON Subject.DepartmentID =
Department.DepartmentID INNER JOIN Student ON StudentGroup.StudentID = Student.StudentID
INNER JOIN DatasetID ON Class.DatasetID = DatasetID.DatasetID WHERE Class.ClassID = '" &

```

```

ClassID & "' and DatasetID.DatasetID = '" & moduleDataset & "' ORDER BY
Student.Firstname ASC, Student.Surname ASC"

    'the dataset is set after passing the following SQL statement and table into
    the GenerateDataset sub of the public module
    ds = generateDataset(sql, "temp")
    'sets the datasource of the datagrid
    Viewstudentsgrid.DataSource = ds
    'sets the following datagrids as either visible or not visible
    Classesgrid.Visible = False
    Studentclassesgrid.Visible = False
    Viewstudentsgrid.Visible = True
    'sets the datamember of the datagrid
    Viewstudentsgrid.DataMember = "temp"
    'makes the ClassID column not visible to the user
    Viewstudentsgrid.Columns("ClassID").Visible = False

    'creates a new variable that stores information for a button column that can
    be added to the datagrid
    Dim deleteStudentColumn As New DataGridViewButtonColumn
    With deleteStudentColumn
        'sets the header text of the button column
        .HeaderText = "Remove this Student from this class"
        'sets the name of the button column
        .Name = "Details"
        'sets the text displayed on each button of the button column
        .Text = "Remove Student"
        .UseColumnTextForButtonValue = True
    End With

    'adds the button column to the datagrid
    Viewstudentsgrid.Columns.Add(deleteStudentColumn)
    'automatically resizes all column widths in the datagrid
    Viewstudentsgrid.AutoResizeColumns()
End If

    'if the selected index of the button that is clicked is the eighth column then the
    following action is performed
    If e.ColumnIndex = 8 Then
        Add_Student_Class.Close()
        'returns the value of the ClassID that is on the same row as the button
        clicked and is under the column with the header of classID
        ClassID = Classesgrid.Rows(e.RowIndex).Cells("ClassID").Value

        'sets the ClassID of the add student Class form as the ClassID of this form
        Add_Student_Class.ClassID = ClassID
        Add_Student_Class.Show()
    End If
End Sub

    'when the text within the studentclassesfield combobox is changed then the code within
    this sub is run
    Private Sub studentclassesfield_SelectedIndexChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles studentclassesfield.SelectedIndexChanged
        'stores dt as a datatable
        Dim dt As New DataTable
        'sets the wherelabel as visible to the user
        Wherelabel1.Visible = True

        'This case statement selects which textbox or combobox is visible when the
        corresponding text value of the field combobox is selected and sets all other textboxes or
        checkboxes as not visible when a case is selected by the user's mouse click choice
        Select Case studentclassesfield.Text

```

```

Case ""
    SearchGender.Visible = False
    SearchName.Visible = False
    SearchForm.Visible = False
    SearchYear.Visible = False
    Studentclassesgrid.Visible = False
    Wherelabel1.Visible = False
Case "Name"
    SearchGender.Visible = False
    SearchName.Visible = True
    SearchForm.Visible = False
    SearchYear.Visible = False
    Wherelabel1.Text = "Name:"
Case "Gender"
    'Uses the Generate combo function of the module to populate a combobox by
    passing the table, field and needempty variables to the Generatecombo function and then
    returning the result in a datatable
    dt = GenerateCombo("Student", "Gender", False)
    'sets the datasource of the combobox
    SearchGender.DataSource = dt
    'sets the display member of the combobox
    SearchGender.DisplayMember = "Gender"

    SearchGender.Visible = True
    SearchName.Visible = False
    SearchForm.Visible = False
    SearchYear.Visible = False
    Wherelabel1.Text = "Gender:"
Case "Form"
    'selects the forms based on the user's choice of year
    sql = "SELECT Distinct Form FROM Student WHERE DatasetID = '' &
moduleDataset & """
    'stores a SQL command that uses the SQL statement to query the database
    that is found when following the connection string of con
    Using comm As SqlCommand = New SqlCommand(sql, con)
        'stores a SQL dataadareader which can read the contents of the sql
        query from the database
        Dim rs As SqlDataReader = comm.ExecuteReader
        'stores a datatable
        'the datatable is filled with the values that the datareader reads
        dt.Load(rs)

        'sets the displaymembe of the combobox
        SearchForm.DisplayMember = "Form"
        'sets the datasource of the combobox
        SearchForm.DataSource = dt
End Using

    SearchGender.Visible = False
    SearchName.Visible = False
    SearchForm.Visible = True
    SearchYear.Visible = False
    Wherelabel1.Text = "Form:"
Case "Year"
    dt = GenerateCombo("Student", "Year", False)
    SearchYear.DataSource = dt
    SearchYear.DisplayMember = "Year"

    SearchGender.Visible = False
    SearchName.Visible = False
    SearchForm.Visible = False
    SearchYear.Visible = True

```

```

        Wherelabel1.Text = "Year:"
    End Select
End Sub
'this sub runs the following code when the search students button is clicked
Private Sub Searchstudentclasses_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Searchstudentclasses.Click
    'clears the current datagrid column selection
    Studentclassesgrid.Columns.Clear()
    'sets the two other datagrids visible state to false
    Classesgrid.Visible = False
    Viewstudentsgrid.Visible = False
    'stores ds as a dataset variable
    Dim ds As DataSet

    'sets the SQL statemnt
    sql = " Select StudentID, Firstname, Surname, Gender, DateOfBirth, Form, Year FROM
Student "
    'This Case statement is used to alter the SQL query depending on which text is
selected from the combobox, when a case is selected the above SQL statement and the
corresponding case 'SQL where statement' are concatenated to form one SQL statement
    Select Case studentclassesfield.Text
        Case ""
            sql = sql & " WHERE DatasetID = '" & moduleDataset & "' ORDER BY
StudentID ASC"
        Case "Name"
            sql = sql & " WHERE (Firstname LIKE '%" & SearchName.Text & "%' Or Surname
LIKE '%" & SearchName.Text & "%') and DatasetID = '" & moduleDataset & "' ORDER BY
StudentID ASC "
        Case "Gender"
            sql = sql & " WHERE Gender LIKE '%" & SearchGender.Text & "%' and
DatasetID = '" & moduleDataset & "' ORDER BY StudentID ASC"
        Case "Form"
            sql = sql & " WHERE Form LIKE '%" & SearchForm.Text & "%' and DatasetID =
'" & moduleDataset & "' ORDER BY StudentID ASC"
        Case "Year"
            sql = sql & " WHERE Year LIKE '%" & SearchYear.Text & "%' and DatasetID =
'" & moduleDataset & "' ORDER BY StudentID ASC"
    End Select

    'the dataset is set after passing the following SQL statement and table into the
GenerateDataset sub of the public module
    ds = generateDataset(sql, "Student")
    'sets the datasource of the datagrid
    Studentclassesgrid.DataSource = ds
    'sets the datagrid as visible
    Studentclassesgrid.Visible = True
    'sets the datamember of the datagrid
    Studentclassesgrid.DataMember = "Student"
    'hides the StudentID column from the user
    Studentclassesgrid.Columns("StudentID").Visible = False

    'creates a new variable that stores information for a button column that can be
added to the datagrid
    Dim Column As New DataGridViewButtonColumn
    With Column
        'sets the header text of the button column
        .HeaderText = "View a Students Classes"
        'sets the name of the button column
        .Name = "Details"
        'sets the text of the button column
        .Text = "View Classes"
        .UseColumnTextForButtonValue = True
    End With

```

```

End With

'adds the button column to the studentclasses datagrid
Studentclassesgrid.Columns.Add(Column)
'autoresizes all columns in the datagrid
Studentclassesgrid.AutoResizeColumns()

'if there are not any rows returned in thedatagrid then the user is asked to
enter a valid name and the datagrid is not visible
If (Studentclassesgrid.Rows.Count = 0) Then
    MsgBox("Please enter a valid name")
    Studentclassesgrid.Visible = False
End If
End Sub

Private Sub studentsClasses_CellContentClick(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.DataGridViewCellEventArgs) Handles
Studentclassesgrid.CellContentClick
    'stores ds as a dataset
    Dim ds As New DataSet
    'stores the studentID
    Dim studentid As String
    If e.ColumnIndex = 7 Then
        'the StudentID variable is set to the StudentID of the StudentID that is on
        the same row as the button that is clicked
        studentid = Studentclassesgrid.Rows(e.RowIndex).Cells("StudentID").Value
        'clears the currentdatagrid column selection
        Studentclassesgrid.Columns.Clear()
        'sets the SQL statement
        sql = "SELECT Class.ClassID, Student.Firstname, Student.Surname,
Class.ClassGroup, Subject.Subject, Department.Department, DatasetID.Year FROM StudentGroup
INNER JOIN Class ON StudentGroup.ClassID = Class.ClassID INNER JOIN Subject ON
Class.SubjectID = Subject.SubjectID INNER JOIN Department ON Subject.DepartmentID =
Department.DepartmentID INNER JOIN Student ON StudentGroup.StudentID = Student.StudentID
INNER JOIN DatasetID ON Class.DatasetID = DatasetID.DatasetID WHERE Student.StudentID = ''
& studentid & '' ORDER BY Class.ClassID ASC"
        'the dataset is set after passing the following SQL statement and table into
        the GenerateDataset sub of the public module
        ds = generateDataset(sql, "temp")
        'sets the datasource of thedatagrid
        Studentclassesgrid.DataSource = ds
        'sets the datamember of thedatagrid
        Studentclassesgrid.DataMember = "temp"
    End If
End Sub

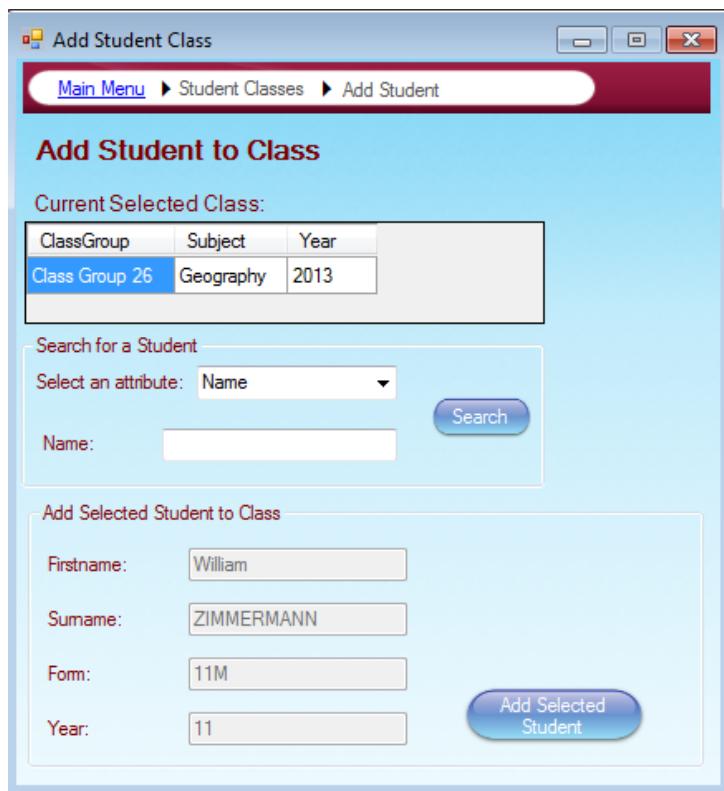
'deals with the print form button
Private Sub printbutton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles printbutton.Click
    'sets the printer settings to default
    PrintForm.PrinterSettings = PrintDialog.PrinterSettings
    If PrintDialog.ShowDialog() = DialogResult.OK Then
        'allows the user to change print settings
        PrintForm.PrinterSettings = PrintDialog.PrinterSettings
        'changes the orientation of the printed document to landscape
        Me.PrintForm.PrinterSettings.DefaultPageSettings.Landscape = True
        'refreshes the form so that the printdialog is not printed along with the form
        Me.Refresh()
        'prints the current form
        Me.PrintForm.Print()
        'Percentageschart.Printing.Print(True)
    End If

```

End Sub
End Class

Add Student Class

This form allows a user to add a student to a class. The class that the user has selected displays in a data grid so that the user is aware of the class they will be adding a student to and can check whether they have selected the incorrect class. To add a student to a class the user must first search for a student and select that student, which they can do so using filter options. The details of the selected student are then displayed to the user in a series of textbox fields and a new button appears which allows the user to add the selected student to the current selected class.



```

Add_Student_Class System.Windows.Forms.Form
addstudent System.Windows.Forms.Button
addstudentgroup System.Windows.Forms.GroupBox
addstudentlink System.Windows.Forms.LinkLabel
banner System.Windows.Forms.PictureBox
Classeslink System.Windows.Forms.LinkLabel
Currentclassgrid System.Windows.Forms.DataGridView
Displaystudent System.Windows.Forms.Button
Field System.Windows.Forms.ComboBox
fieldlabel System.Windows.Forms.Label
Firstnamefield System.Windows.Forms.TextBox
firstnamelabel System.Windows.Forms.Label
Formfield System.Windows.Forms.TextBox
formheader System.Windows.Forms.Label
formlabel System.Windows.Forms.Label
Gridheaderlabel System.Windows.Forms.Label
linkarrow1 System.Windows.Forms.Label
linkarrow2 System.Windows.Forms.Label
MainMenu System.Windows.Forms.LinkLabel
SearchDataset System.Windows.Forms.ComboBox
SearchEthnicity System.Windows.Forms.ComboBox
SearchForm System.Windows.Forms.ComboBox
SearchGender System.Windows.Forms.ComboBox
Searchgroup System.Windows.Forms.GroupBox
SearchName System.Windows.Forms.TextBox
SearchSNS System.Windows.Forms.ComboBox
SearchYear System.Windows.Forms.ComboBox
Studentdisplaygrid System.Windows.Forms.DataGridView
Surnamefield System.Windows.Forms.TextBox
surnamelabel System.Windows.Forms.Label

```

Parameter Name	Description
ClassID	This sub uses a get command to fetch the corresponding ClassID of the row the user wants to add a student to and stores it as a private integer, which will be used to reference the correct class row in the database.
MainMenu_LinkClicked	When the main menu link label is clicked all open forms close and the main menu displays.
Classeslink_LinkClicked	When the Student Classes link label is clicked all this form closes and the Student Classes form displays.
Addstudentclass_Load	When the form initially loads the Class ID that is passed to this form is used to search for the class record matching the Class ID in the database. The result of this query is used to display the details of this class in the current class's data grid.
Field_SelectedIndexChanged	Whenever the text value of the filter field is changed this sub determines which objects are either visible or non-visible.
Displaystudent_Click	This sub runs when the Search button is clicked and returns a query matching the criteria the user has used to filter their Student selection by, this selection is then displayed in the data grid.
validatestudentclass	This function checks to see whether the student that has been selected is already in the class that the user wants to add the student to. A true or false condition is returned following this validation.
StudentGrid_CellClick	When one of the select buttons is clicked on the data grid the details of the selected student on that row are set as the text values of their corresponding textboxes. This is only done though after checking the student is not already in the selected class.
addstudent_Click	When the button that adds the student to the selected class is clicked a new StudentGroupID is dynamically created which is set as the next available ID. A new Student Group record is then inserted into the database using a SQL insert statement that consists of the new ID and the values of the textboxes (which consist of a student's details). The form then closes.

```

Imports the following system code commands so that I can use them appropriately
Imports System.Data.SqlClient
Imports System.Data
Imports System.Configuration
Imports System.Data.DataTable
Public Class Add_Student_Class
    'sets the variable that is received from the StudentClass form as a private variable
    which can be used in any sub within this class
    Private _ClassID As Integer
    'stores a studentID variable which can only be used within this class
    Dim studentid As Integer
    'creates a property of the variable that is being received
    Public Property ClassID() As Integer
        Get
            'gets the variable from the form that has been specified, in this case the
            grade form
            Return _ClassID
        End Get
        Set(ByVal value As Integer)
            'sets the variable that has been received by the get statement as the
            equivalent variable that will be used in this form
            _ClassID = value
        End Set
    End Property
    'this sub runs when the main menu link is clicked by the user
    Private Sub MainMenu_LinkClicked(sender As System.Object, e As
System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles MainMenu.LinkClicked
        'the following forms either display or close
        Main_Menu.Show()
        Student_Classes.Close()
        Remove_Student_Class.Close()
        Myclasses.Close()
        Me.Close()
    End Sub
    'this sub runs when the Student Classes link is clicked by the user
    Private Sub Classeslink_LinkClicked(sender As System.Object, e As
System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles Classeslink.LinkClicked
        'the following forms either display or close
        Student_Classes.Show()
        Me.Close()
    End Sub
    'this sub runs when the form is initially loaded
    Private Sub Addstudentclass_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Me.Load
        'clears the current selection of columns in the datagrid
        Currentclassgrid.Columns.Clear()
        'stores ds as a dataset
        Dim ds As New DataSet

        'sets the SQL statement which queries the database
        sql = "SELECT DISTINCT Class.ClassGroup, Subject.Subject, DatasetID.Year FROM
Department INNER JOIN Subject ON Department.DepartmentID = Subject.DepartmentID INNER JOIN
Class INNER JOIN DatasetID ON Class.DatasetID = DatasetID.DatasetID ON Subject.SubjectID =
Class.SubjectID INNER JOIN Teacher INNER JOIN Teaches ON Teacher.TeacherID =
Teaches.TeacherID ON Class.TeacherID = Teacher.TeacherID WHERE Class.ClassID = " & ClassID
& ""

        'the dataset is set after passing the following SQL statement and table into the
        GenerateDataset sub of the public module
        ds = generateDataset(sql, "temp")
        'sets the datasource of the datagrid
        Currentclassgrid.DataSource = ds
        'sets the datagrid as visible
    End Sub

```

```

        Currentclassgrid.Visible = True
        'sets the datamember of the datagrid
        Currentclassgrid.DataMember = "temp"
        'auto resizes the columns within the datagrid
        Currentclassgrid.AutoResizeColumns()

    End Sub
    'when the text within the field combobox is changed then the code within this sub is
run
    Private Sub Field_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Field.SelectedIndexChanged
        'stores dt as a datatable

        'sets the wherelabel as visible to the user
        wherelabel.Visible = True

        'This case statement selects which textbox or combobox is visible when the
corrsponding text value of the field combobox is selected and sets all other textboxes or
comboboxes as not visible when a case is selected by the user's mouse click choice
        Select Case Field.Text
            Case ""
                SearchName.Visible = False
                SearchForm.Visible = False
                Displaystudent.Visible = False
                wherelabel.Visible = False
            Case "Name"
                SearchName.Visible = True
                SearchForm.Visible = False
                Displaystudent.Visible = True
                wherelabel.Text = "Name:"
            Case "Form"
                'selects the forms based on the user's choice of year
                sql = "SELECT Distinct Form FROM Student WHERE DatasetID = " &
moduleDataset & "'"
                'stores a SQL command that uses the SQL statement to query the database
                that is found when following the connection string of con
                Using comm As SqlCommand = New SqlCommand(sql, con)
                    'stores a SQL dataadareader which can read the contents of the sql
query from the database
                    Dim rs As SqlDataReader = comm.ExecuteReader
                    'stores a datatable
                    Dim dt As DataTable = New DataTable
                    'the datatable is filled with the values that the datareader reads
                    dt.Load(rs)

                    'sets the displaymemeber of the combobox
                    SearchForm.DisplayMember = "Form"
                    'sets the datasource of the combobox
                    SearchForm.DataSource = dt
                End Using

                SearchName.Visible = False
                SearchForm.Visible = True

                Displaystudent.Visible = True
                wherelabel.Text = "Form:"

            End Select
        End Sub
        'this sub runs when the search button is clicked by the user
        Private Sub Displaystudent_Click(sender As System.Object, e As System.EventArgs)
Handles Displaystudent.Click

```

```

'clears the current selection of columns in the studentgrid
Studentdisplaygrid.Columns.Clear()
'stores a dataset variable
Dim ds As DataSet

'sets the SQL statement
sql = " Select Firstname, Surname, Form, Year FROM Student "
'This Case statement is used to alter the SQL query depending on which text is
selected from the combobox, when a case is selected the above SQL statement and the
corresponding case 'SQL where statement' are concatenated to form one SQL statement
Select Case Field.Text
    Case "Name"
        sql = sql & " WHERE (Firstname LIKE '%" & SearchName.Text & "%' Or Surname
LIKE '%" & SearchName.Text & "%') and DatasetID = '" & moduleDataset & "' ORDER BY
StudentID ASC "
    Case "Form"
        sql = sql & " WHERE Form LIKE '%" & SearchForm.Text & "%' and DatasetID =
'" & moduleDataset & "' ORDER BY StudentID ASC"
    Case "Year"
        sql = sql & " WHERE Year LIKE '%" & SearchYear.Text & "%' and DatasetID =
'" & moduleDataset & "' ORDER BY StudentID ASC"
End Select

'the dataset is set after passing the following SQL statement and table into the
GenerateDataset sub of the public module
ds = generateDataset(sql, "Student")
'sets the datasrouce of the datagrid
Studentdisplaygrid.DataSource = ds
'sets the following datagrids as either visible or false
addstudentgroup.Visible = False
Studentdisplaygrid.Visible = True
'sets the datamember of the datagrid
Studentdisplaygrid.DataMember = "Student"

'if there is only one row in the datagrid then the entered username is not a valid
username and so the following error message is displayed to the user and the grid is not
visible
If (Studentdisplaygrid.Rows.Count = 0) Then
    MsgBox("Please enter a valid name")
    Studentdisplaygrid.Visible = False
End If

'creates a new variable that stores information for a button column that can be
added to the datagrid
Dim StudentColumn As New DataGridViewButtonColumn
With StudentColumn
    'sets the header text of the button column
    .HeaderText = "Select Student"
    'sets the name of the button column
    .Name = "Details"
    'sets the text displayed on the buttons of the button column in the datagrid
    .Text = "Select"
    .UseColumnTextForButtonValue = True
End With

'adds the button column to the datagrid
Studentdisplaygrid.Columns.Add(StudentColumn)
'auto resizes the column widths of the datagrid
Studentdisplaygrid.AutoResizeColumns()
End Sub
'this function returns a true or false state depending on whether the student selected
is a part of the current selected class

```

```

Public Function validatestudentclass(ByVal studentid As Integer) As Boolean
    'stores cmd as a sqlCommand()
    Dim Cmd As New SqlCommand
    'stores da as a SQL dataadapter
    Dim da As New SqlDataAdapter
    'stores dt as a datatable
    Dim dt As New DataTable
    'stores ds as a dataset
    Dim ds As New DataSet
    'connects to the SQL database by calling the DatabaseConnect from the module
    Call DatabaseConnect()

    'sets the SQL statement
    sql = "Select * FROM StudentGroup WHERE StudentID = '" & studentid & "' AND
    ClassID = '" & ClassID & "'"
    'stores a SQL command that uses the SQL statement to query the database that is
    found when following the connection string of con
    Dim cmmnd As New SqlCommand(sql, con)
    'the dataadapter then uses this SQL command to Query the database
    da.SelectCommand = cmmnd
    'the dataadapter retrieves the query results from the SQL statement after
    connecting to the Database and fills a dataset
    da.Fill(ds, "temp")
    'the datatable variable is then set to the table result of the dataset variable
    dt = ds.Tables("temp")
    'if the nubmer of rows in the datatable is one or more then the IDexists state is
    set to false otherwise it is set to true
    If dt.Rows.Count >= 1 Then
        validatestudentclass = False
    Else
        validatestudentclass = True
    End If
End Function
'this sub runs when one of the buttons that is part of the button column on the
datagrid is clicked
Private Sub StudentGrid_CellClick(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.DataGridViewCellEventArgs) Handles Studentdisplaygrid.CellClick
    'stores variables to identify the student
    Dim firstname As String
    Dim surname As String
    Dim form As String
    Dim year As String

    Dim Cmd As New SqlCommand
    Dim da As New SqlDataAdapter
    Dim dt As New DataTable
    Dim ds As New DataSet

    'if the cell clicked is of index 4 then the following code runs
    If e.ColumnIndex = 4 Then
        'the student's information variables are set as their corrsponding values
        under their respective headings on the same row as the button clicked
        firstname = Studentdisplaygrid.Rows(e.RowIndex).Cells("Firstname").Value
        surname = Studentdisplaygrid.Rows(e.RowIndex).Cells("Surname").Value
        form = Studentdisplaygrid.Rows(e.RowIndex).Cells("Form").Value
        year = Studentdisplaygrid.Rows(e.RowIndex).Cells("Year").Value

        'this block of code sets the StudentID of the record that the user wants to
        add based on the values of the student information variables
        sql = "Select DISTINCT StudentID FROM Student WHERE Firstname = '" & firstname
        & "' AND Surname = '" & surname & "' AND Form = '" & form & "' AND Year = '" & year & "' "
        Dim cmmnd As New SqlCommand(sql, con)

```

```

da.SelectCommand = cmmnd
da.Fill(ds, "Student")
dt = ds.Tables("Student")
studentid = ds.Tables(0).Rows(0).Item("StudentID").ToString()

'if the student is not a part of the current selected class then the following
code runs
If validatestudentclass(studentid) = True Then

    'the values of the textbox fields are set as there respective variables
    Firstnamefield.Text = firstname
    Surnamefield.Text = surname
    Formfield.Text = form
    Yearfield.Text = year

    'the datagrid is set to be no longer visible and the add group becomes
visible
    addstudentgroup.Visible = True
    Studentdisplaygrid.Visible = False
Else
    'if the student is in the currently selected class then the following
error message is displayed and nothing happens
    MsgBox("That Student is already in the Selected Class",
MsgBoxStyle.Information, MessageBoxButtons.OK)
End If

End If

End Sub
'this sub runs when the add button is clicked by the user
Private Sub addstudent_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles addstudent.Click
    Dim da As New SqlDataAdapter
    Dim dt As New DataTable
    Dim ds As New DataSet
    'stores the studentgroupid of the record that will be inserted
    Dim StudentGroupID As Integer

    'sets the SQL satatement
    sql = "Select TOP 1 StudentGroupID FROM StudentGroup ORDER BY StudentGroupID DESC"
    'stores cmmnd as a SQL command that uses the above SQL statement and the connection
string of con
    Dim cmmnd As New SqlCommand(sql, con)
    'sets the dataadapter select command as cmmnd
    da.SelectCommand = cmmnd
    'the dataadapter retrieves the query results from the SQL statement after
connecting to the Database and fills a dataset
    da.Fill(ds, "studentgroup")
    'the datatable variable is then set to the table result of the dataset variable
    dt = ds.Tables("studentgroup")
    'The StudentgroupID is then set as the value of the highest studentgroupID found
from the Query and then incremented by 1, the SQL statement takes into account removing
students from classes so that inserting a record will use the first available
studentgroupID
    StudentGroupID = ds.Tables(0).Rows(0).Item("StudentGroupID").ToString() + 1

    DatabaseConnect()
    'sets the SQL statement
    sql = "INSERT INTO StudentGroup (StudentGroupID, ClassID, StudentID) VALUES
(@StudentGroupID, @ClassID, @StudentID)"
    'uses cmd as a new SQL command that uses the above SQL statement and the
connection string of con

```

```
Using cmd = New SqlCommand(sql, con)
    'sets the parameters of the vlues to be inserted
    cmd.Parameters.AddWithValue("StudentGroupID", StudentGroupID)
    cmd.Parameters.AddWithValue("ClassID", ClassID)
    cmd.Parameters.AddWithValue("StudentID", studentid)
    cmd.ExecuteNonQuery()
End Using
'closes the database connection
con.Close()
'notifies the user with the appropriate message
MsgBox("Record Added", MsgBoxStyle.Information)
'closes this form
Me.Close()

End Sub
End Class
```

Remove Student Class

From this form a user can delete a student from the selected class; the selected class is determined by the Class ID that is passed to this form. When the Class ID is received by the form, the database is queried using the Class ID and then the Data grid is filled with the details of the class that is returned from the query. The details of the student that the user wants to remove from the class are also passed into the form. These values are used to fill the textbox fields with the corresponding Student's details so that the user can see which student they are about to remove from which class. As such this acts as a verification method. To remove the selected student from the selected class the user must click the remove from class button.

The screenshot shows a Windows application window titled "Remove Student Class". The title bar includes standard window controls (minimize, maximize, close). Below the title bar is a navigation menu bar with the items "Main Menu", "Student Classes", and "Remove Student". The main content area has a red header bar with the text "Remove Student From Class". Below this, a label "Current Selected Class:" is followed by a DataGridView containing one row of data:

ClassGroup	Subject	Year
Class Group 107	Geography	2014

Below the grid are three text input fields labeled "Student Firstname:", "Student Surname:", and "Student Form:", each with a corresponding text box. At the bottom right is a blue button labeled "Remove from Class".

```

banner System.Windows.Forms.PictureBox
Classeslink System.Windows.Forms.LinkLabel
Currentclassgrid System.Windows.Forms.DataGridView
DatagridHeaderlabel System.Windows.Forms.Label
Firstnamefield System.Windows.Forms.TextBox
firstnamelabel System.Windows.Forms.Label
Formfield System.Windows.Forms.TextBox
formlabel System.Windows.Forms.Label
formtitle System.Windows.Forms.Label
linkarrow1 System.Windows.Forms.Label
linkarrow2 System.Windows.Forms.Label
MainMenulink System.Windows.Forms.LinkLabel
Remove_Student_Class System.Windows.Forms.Form
Removefromclass System.Windows.Forms.Button
removestudentlink System.Windows.Forms.LinkLabel
Surnamefield System.Windows.Forms.TextBox
surnamelabel System.Windows.Forms.Label

```

Parameter Name	Description
Fname	This sub uses a get command to fetch the first name of the student that the user wants to remove.
Form	This sub uses a get command to fetch the form of the student that the user wants to remove.
Sname	This sub uses a get command to fetch the surname of the student that the user wants to remove.
ClassID	This sub uses a get command to fetch the corresponding Class ID of the row the user wants to remove a student from and stores it as a private integer, which will be used to reference the correct class row in the database.
MainMenu_LinkClicked	When the main menu link label is clicked all open forms close and the main menu displays.
Classeslink_LinkClicked	When the Student Classes link label is clicked all this form closes and the Student Classes form displays.
Removestudentclass_Load	When the form initially loads the Class ID that is passed to this form is used to search for the class record matching the Class ID in the database. The result of this query is used to display the details of this class in the current class's data grid. The text properties of the textbox fields are also set as their equivalent values that have been passed into the form.
removefromclass_Click	When the remove from class button is clicked by the user, the student group record that matches the ID of the selected student and the ID of the selected class is deleted from the database. The user is asked if they would like to confirm their choice.

```

'Imports the following system code commands so that I can use them appropriately
Imports System.Data.SqlClient
Imports System.Data
Imports System.Configuration
Imports System.Data.DataTable
Public Class Remove_Student_Class
    'sets the variables that are received from the student classes form as private
    variables which can be used in any sub within this class
    Private _ClassID As Integer
    Private _Fname As String
    Private _Sname As String
    Private _Form As String
    'creates a property of the variable that is being received
    Public Property Fname() As String
        Get
            'gets the variable from the form that has been specified, in this case the
            student classes form
            Return _Fname
        End Get
        Set(ByVal value As String)
            'sets the variable that has been received by the get statement as the
            equivalent variable that will be used in this form
            _Fname = value
        End Set
    End Property
    Public Property Form() As String
        Get
            Return _Form
        End Get
        Set(ByVal value As String)
            _Form = value
        End Set
    End Property
    Public Property Sname() As String
        Get
            Return _Sname
        End Get
        Set(ByVal value As String)
            _Sname = value
        End Set
    End Property
    Public Property ClassID() As Integer
        Get
            Return _ClassID
        End Get
        Set(ByVal value As Integer)
            _ClassID = value
        End Set
    End Property
    'when the main menu link is clicked the following forms either close or open
    Private Sub MainMenu_LinkClicked(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles MainMenulink.LinkClicked
        Main_Menu.Show()
        Add_Student_Class.Close()
        Student_Classes.Close()
        Myclasses.Close()
        Me.Close()
    End Sub
    'when the Classes link is clicked the following forms either close or open
    Private Sub Classeslink_LinkClicked(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles Classeslink.LinkClicked
        Student_Classes.Show()

```

```

        Me.Close()
    End Sub
    'this sub runs when the form initially loads
    Private Sub Removestudentclass_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Me.Load
        'clears the current column selection of the datagrid
        Currentclassgrid.Columns.Clear()
        'stores ds as a dataset
        Dim ds As New DataSet

        'sets the SQL statement which queries the database
        sql = "SELECT DISTINCT Class.ClassGroup, Subject.Subject, DatasetID.Year FROM
Department INNER JOIN Subject ON Department.DepartmentID = Subject.DepartmentID INNER JOIN
Class INNER JOIN DatasetID ON Class.DatasetID = DatasetID.DatasetID ON Subject.SubjectID =
Class.SubjectID INNER JOIN Teacher INNER JOIN Teaches ON Teacher.TeacherID =
Teaches.TeacherID ON Class.TeacherID = Teacher.TeacherID WHERE Class.ClassID = " & ClassID
& ""

        'the dataset is set after passing the following SQL statement and table into the
GenerateDataset sub of the public module
        ds = generateDataset(sql, "temp")
        'sets the datasource of the datagrid
        Currentclassgrid.DataSource = ds
        'sets thedatagrid as visible
        Currentclassgrid.Visible = True
        'sets the datamember of thedatagrid
        Currentclassgrid.DataMember = "temp"
        'auto resizes the columns within thedatagrid
        Currentclassgrid.AutoResizeColumns()

        'sets teh follwoing text fields as the corresponding variables that have been
recieved from the Student classes form
        Firstnamefield.Text = Fname
        Surnamefield.Text = Sname
        Formfield.Text = Form
    End Sub
    'this sub runs the following code when the remove button is clicked
    Private Sub removefromclass_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Removefromclass.Click
        'stores the StudentID
        Dim studentID As String
        'stores da as a SQL data adapter
        Dim da As New SqlDataAdapter
        'stores dt as a datatable
        Dim dt As New DataTable
        'stores ds as a dataset
        Dim ds As New DataSet

        'Calls the database connect sub from the module
        DatabaseConnect()

        'sets the SQL statement which finds the StudentID from the values that have been
entered in the firstname and surname textboxes
        sql = "Select Distinct StudentID FROM Student WHERE Firstname = '" &
Firstnamefield.Text & "' AND Surname = '" & Surnamefield.Text & "' "
        'stores cmmnd as a SQL command that uses the above SQL statement and the connection
string of con
        Dim cmmnd As New SqlCommand(sql, con)
        'sets the dataadapter select command as cmmnd
        da.SelectCommand = cmmnd
        'the dataadapter retrieves the query results from the SQL statement after
connecting to the Database and fills a dataset
        da.Fill(ds, "Student")
    
```

```
'the datatable variable is then set to the table result of the dataset variable
dt = ds.Tables("Student")
'The student ID variable is then set to the first row that is returned in the
dataset after querying the database with the above SQL statement
studentID = ds.Tables(0).Rows(0).Item("StudentID").ToString()

'Stores a yes/no messagebox with the following message
Dim result As Integer = MessageBox.Show("Do you really want to delete this
record?", "Delete", MessageBoxButtons.YesNo, MessageBoxIcon.Warning)
'if the user clicks the yes option when prompted the folliwng runs
If result = DialogResult.Yes Then
    'sets the SQL statemnt
    sql = "DELETE FROM StudentGroup WHERE StudentID = '" & studentID & "' AND
ClassID = '" & ClassID & "'"
    'uses cmmnd as a new SQL command that uses the above SQL statement and the
connection string of con
    Using cmd = New SqlCommand(sql, con)
        'executes the query and deletes the record from the Dataabse
        cmd.ExecuteNonQuery()
    End Using
    con.Close()
    'displays a messagebox to the user to verify that the record has been deleted
from the database
    MsgBox("Record Deleted From Database", MsgBoxStyle.Information)
    'closes this form and displays the student classes form
    Student_Classes.Show()
    Me.Close()
    'but if the user clicks the no option when prompted nothing happens
ElseIf result = DialogResult.No Then
    End If

End Sub
End Class
```

Developer Features

The Developer features form allows a user with administrative privileges to use a set of advanced controls that not all users should have access to. The set of commands that an advanced user can use are as follows, add a new Dataset Year, Add a new Grade Type, Add a new Subject, Add a new Level, Add a new login and Edit an existing login. When an admin selects one of these controls the data grid is populated with the details of the existing contents of that admin action. For example if a user chooses to add a new subject (as seen below) then the data grid will be populated with all existing subjects in the system. From here an existing subject can be deleted from the system and a new subject can be added to the system. A hard copy of any search criteria can also be produced by clicking the print icon.

The screenshot shows the 'Admin Features' application window. At the top left is the title bar 'Admin Features'. Below it is a navigation bar with 'Main Menu' and 'Admin Features'. On the right side of the navigation bar is a camera icon. The main area has a header 'Admin Features' and a sub-header 'Admin Functions'. It contains a dropdown menu 'Choose an Admin Action:' with the option 'Add a new Subject' selected. To the right is a data grid titled 'Record Details' showing a list of subjects with columns for SubjectID, Subject, Department, and Delete Record. At the bottom left is a form for adding a new subject, containing fields for 'New Subject' and 'Department', and buttons for 'Add' and 'View Current Subjects'.

SubjectID	Subject	Department	Record Details
1	Art	Arts	Delete Record
23	Biology	Science	Delete Record
2	Business Studies	Technology	Delete Record
24	Chemistry	Science	Delete Record
22	Computing	Technology	Delete Record
3	Drama	Arts	Delete Record
4	Economics	Humanities	Delete Record
5	English Language	English	Delete Record
6	English Literature	English	Delete Record
7	Food Technology	Technology	Delete Record
14	French	Languages	Delete Record
8	Geography	Humanities	Delete Record
16	German	Languages	Delete Record
17	Graphics	Technology	Delete Record
9	Health And Socia...	Humanities	Delete Record
10	History	Humanities	Delete Record
11	ICT	Technology	Delete Record
12	Leisure and Touri...	Humanities	Delete Record
13	Maths	Maths	Delete Record
19	Musica	Arts	Delete Record

```

Adddatasetyear System.Windows.Forms.Button
Addgradetypebutton System.Windows.Forms.Button
Addlevelbutton System.Windows.Forms.Button
Addloggingroup System.Windows.Forms.GroupBox
Addsubjectbutton System.Windows.Forms.Button
adduser System.Windows.Forms.Button
Adminaction System.Windows.Forms.ComboBox
adminactionlabel System.Windows.Forms.Label
adminfield System.Windows.Forms.ComboBox
adminfield1 System.Windows.Forms.ComboBox
Admingroup System.Windows.Forms.GroupBox
adminlabel System.Windows.Forms.Label
adminlabel1 System.Windows.Forms.Label
banner System.Windows.Forms.PictureBox
Dataset System.Windows.Forms.GroupBox
Datasetsyearfield System.Windows.Forms.TextBox
Departmentfield System.Windows.Forms.ComboBox
Departmentlabel System.Windows.Forms.Label
Developer System.Windows.Forms.LinkLabel
Developer_Features System.Windows.Forms.Form
editlogin System.Windows.Forms.Button
Editloggingroup System.Windows.Forms.GroupBox
edittitle System.Windows.Forms.Label
Gradetypegroup System.Windows.Forms.GroupBox
Levelfield System.Windows.Forms.TextBox
Levelgroup System.Windows.Forms.GroupBox
linkarrow System.Windows.Forms.Label
loginidfield System.Windows.Forms.TextBox
loginidlabel System.Windows.Forms.Label
MainMenu System.Windows.Forms.LinkLabel

```

```

Newdatasetsyearlabel System.Windows.Forms.Label
Newgradetypefield System.Windows.Forms.TextBox
Newgradetypelabel System.Windows.Forms.Label
Newlevellabel System.Windows.Forms.Label
Newsubjectlabel System.Windows.Forms.Label
passwordfield System.Windows.Forms.TextBox
passwordfield1 System.Windows.Forms.TextBox
passwordlabel System.Windows.Forms.Label
passwordlabel1 System.Windows.Forms.Label
printbutton System.Windows.Forms.Button
PrintDialog System.Windows.Forms.PrintDialog
PrintForm Microsoft.VisualBasic.PowerPacks.Printing.PrintForm
Subjectgroup System.Windows.Forms.GroupBox
usernamefield System.Windows.Forms.TextBox
usernamefield1 System.Windows.Forms.TextBox
usernamelabel System.Windows.Forms.Label
usernamelabel1 System.Windows.Forms.Label
Viewcurrentdatasetsyear System.Windows.Forms.Button
Viewcurrentgradetype System.Windows.Forms.Button
Viewcurrentlevel System.Windows.Forms.Button
Viewcurrentsubjects System.Windows.Forms.Button
Viewcurrentvalues System.Windows.Forms.DataGridView
vieweditlogins System.Windows.Forms.Button
viewlogins System.Windows.Forms.Button

```

Parameter Name	Description
CurrentvaluesGrid_ContentClick	This sub handles the instance one of the buttons That is a part of the view current values grid is clicked, depending on the selection within the data grid the correct delete routine is then run.
DeveloperFeatures	This handles the instance the form loads and is responsible for connecting to the database along with populating any dynamic combo boxes on the form.
MainMenu_LinkClicked	When this link label is clicked the user is redirected to the Main menu form. If fields on the form have been edited then the user is asked to confirm their navigation choice.
editedcheck	This function sets a true or false condition depending on whether the values in any of the entry boxes differ from their original values. If they differ then a false state is returned.
Viewcurrentsubjects_Click	This routine finds all subjects within the database and fills a dataset with all of the distinct queried results. It then ensures that the data grid displays all the data within the dataset. A button column is added to the data grid allowing the deletion of a record.
Viewcurrentlevel_Click	This routine finds all levels within the database and fills a dataset with all of the distinct queried results. It then ensures that the data grid displays all the data within the dataset. A button column is added to the data grid allowing the deletion of a record.
Viewcurrentdatasetyear_Click	This routine finds all years within the database and fills a dataset with all of the distinct queried results. It then ensures that the data grid displays all the data within the dataset. A button column is added to the data grid allowing the deletion of a record.
viewlogins_Click	This routine finds all users within the database and fills a dataset with all of the distinct queried results. It then ensures that the data grid displays all the data within the dataset. A button column is added to the data grid allowing the deletion of a record.
vieweditlogins_Click	This routine finds all users within the database and fills a dataset with all of the distinct queried results. It then ensures that the data grid displays all the data within the dataset. A button column is added to the data grid allowing the editing of a record.
Viewcurrentgradetype_Click	This routine finds all grade types within the database and fills a dataset with all of the distinct

	queried results. It then ensures that the data grid displays all the data within the dataset. A button column is added to the data grid allowing the deletion of a record.
txtvalidatesubject	This function sets a true or false condition depending on whether any textboxes within the subject group are blank. It also sets the background colour of textboxes appropriately to indicate to the user that fields have not been completed.
txtvalidatelevel	This function sets a true or false condition depending on whether any textboxes within the level group are blank. It also sets the background colour of textboxes appropriately to indicate to the user that fields have not been completed.
txtvalidatedataset	This function sets a true or false condition depending on whether any textboxes within the dataset group are blank. It also sets the background colour of textboxes appropriately to indicate to the user that fields have not been completed.
txtvalidategradetype	This function sets a true or false condition depending on whether any textboxes within the grade type group are blank. It also sets the background colour of textboxes appropriately to indicate to the user that fields have not been completed.
txtvalidatelogin	This function sets a true or false condition depending on whether any textboxes within the login group are blank. It also sets the background colour of textboxes appropriately to indicate to the user that fields have not been completed.
Addsubjectbutton_Click	When the add button that is part of the subject group is clicked a new Subject ID is dynamically created which is set as the next available ID. A new Subject record is then inserted into the database using a SQL insert statement that consists of the new ID and the values of the textboxes.
Addlevelbutton_Click	When the add button that is part of the level group is clicked a new Levelled is dynamically created which is set as the next available ID. A new level record is then inserted into the database using a SQL insert statement that consists of the new ID and the values of the textboxes.
Adddatasetyear_Click	When the add button that is part of the dataset group is clicked a new Dataset ID is dynamically created which is set as the next available ID. A new Dataset record is then inserted into the database using a SQL insert statement that

	consists of the new ID and the values of the textboxes.
Addgradetypebutton_Click	When the add button that is part of the grade type group is clicked a new GradeTypeID is dynamically created which is set as the next available ID. A new grade type record is then inserted into the database using a SQL insert statement that consists of the new ID and the values of the textboxes.
adduser_Click	When the add button that is part of the User group is clicked a new Login ID is dynamically created which is set as the next available ID. A new login record is then inserted into the database using a SQL insert statement that consists of the new ID and the values of the textboxes.
adminaction_SelectedIndexChanged	Whenever the text value of the filter field is changed this sub determines which objects are either visible or non-visible.
editlogin_Click	When the Edit login button is clicked the details of the login record within the database will be updated using a SQL update statement with the values the user inputs within the textboxes.
deletesubject	This sub deals with deleting a subject record from the system. The record that is deleted is the record on the same row as the button that is clicked by the user. A dialog option displays to confirm the user's choice of deletion.
deletedataset	This sub deals with deleting a Dataset record from the system. The record that is deleted is the record on the same row as the button that is clicked by the user. A dialog option displays to confirm the user's choice of deletion.
deleteuser	This sub deals with deleting a user record from the system. The record that is deleted is the record on the same row as the button that is clicked by the user. A dialog option displays to confirm the user's choice of deletion.
deletetradegradetype	This sub deals with deleting a grade type record from the system. The record that is deleted is the record on the same row as the button that is clicked by the user. A dialog option displays to confirm the user's choice of deletion.
edituser	if the button that is clicked has the text 'Edit Record' within it then this sub runs, it gets the values of the user logins details on the same row as the button that is clicked and then uses these values to fill a series of textbox fields which the user can edit.

deletelevel	This sub deals with deleting a level record from the system. The record that is deleted is the record on the same row as the button that is clicked by the user. A dialog option displays to confirm the user's choice of deletion.
printbutton_Click	This sub allows the user to print a hard copy of the form when the print icon is clicked, additional print dialog options are then displayed allowing the user to choose printer and number of copies amongst other settings.

```
'Imports the following system code commands so that I can use them appropriately
Imports System.Data.SqlClient
Imports System.Data
Imports System.Configuration
Imports System.Data.OleDb
Public Class Developer_Features
    'stores a series of check variables which store a string
    Dim subjectcheck As String
    Dim departmentcheck As String
    Dim levelcheck As String
    Dim yearcheck As String
    Dim gradetypecheck As String

    'stores a series of validation variables that can either be true or false
    Dim subjectdelete As Boolean = False
    Dim GradeTypeDelete As Boolean = False
    Dim LevelDelete As Boolean = False
    Dim YearDelete As Boolean = False
    Dim userdelete As Boolean = False
    Dim useredit As Boolean = False

    'this sub runs when a cell that is part of the datagrid is clicked by the user
    Private Sub CurrentvaluesGrid_ContentClick(ByVal sender As System.Object, ByVal e As System.Windows.Forms.DataGridViewCellEventArgs) Handles Viewcurrentvalues.CellClick

        'if the subjectdelete state is true then the row that the button clicked by
        'the user is removed from the database
        If subjectdelete = True Then
            'sets the subjectdelete state back to false
            subjectdelete = False
            deletesubject(e)
        End If

        If LevelDelete = True Then
            LevelDelete = False
            deletelevel(e)
        End If

        If GradeTypeDelete = True Then
            GradeTypeDelete = False
            deletegradetype(e)
        End If

        If useredit = True Then
            edituser(e)
        End If

        If userdelete = True Then
            userdelete = False
            deleteuser(e)
        End If

        If YearDelete = True Then
            YearDelete = False
            deletedataset(e)
        End If
    End Sub

    'this sub runs when the form initially loads
    Private Sub DeveloperFeatures(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Me.Load
        'stores dt as a datatable
```

```

Dim dt As New DataTable
'runs the databaseconnect sub from the public module
DatabaseConnect()

'Uses the Generate combo function of the module to populate a combobox by passing
the table, field and needempty variables to the Generatecombo function and then returning
the result in a datatable
    dt = GenerateCombo("Department", "DepartmentID, Department", True)
    'sets the datasource of the combobox
    Departmentfield.DataSource = dt
    'sets the display member of the combobox
    Departmentfield.DisplayMember = "Department"
    'the index of each item in the combobox is set as their corresponding ID from the
database
    Departmentfield.ValueMember = "DepartmentID"
    'datagrid is not visible
    Viewcurrentvalues.Visible = False

    If addnewdataset = True Then
        Adminaction.Text = "Add a new Dataset Year"
    End If
End Sub
'this sub runs when the main menu link is clicked
Private Sub MainMenu_LinkClicked(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles MainMenu.LinkClicked
    'if the value of the editedcheck function is true the following forms either
display or close
    If editedcheck() = True Then
        Main_Menu.Show()
        Me.Close()
        'if the value of the editedcheck function is false the following messagebox
warning displays to the user, if they select yes the following forms close or open
    ElseIf editedcheck() = False Then
        Dim result As Integer = MessageBox.Show("Do you really want to change
window?", "Edited Record", MessageBoxButtons.YesNo, MessageBoxIcon.Warning)
        If result = DialogResult.Yes Then
            Main_Menu.Show()
            Me.Close()
            'if the user selects no then nothing happens
        ElseIf result = DialogResult.No Then
            End If
        End If
    End If
End Sub
'this function stores an editedcheck variable as either true or false
Public Function editedcheck() As Boolean
    'calls the databaseconnect sub from within the public module
    Call DatabaseConnect()
    'the state of this function is initially set to true
    editedcheck = True
    'if any of the following fields are not empty then the state of this sub changes
to false
    If Newsubjectfield.Text <> "" Then
        editedcheck = False
    End If
    If Departmentfield.Text <> "" Then
        editedcheck = False
    End If
    If Datasetyearfield.Text <> "" Then
        editedcheck = False
    End If
    If Levelfield.Text <> "" Then
        editedcheck = False
    End If

```

```

End If
If Newgradetypefield.Text <> "" Then
    editedcheck = False
End If
End Function
'this sub runs when the viewcurrentsubjects button is clicked by the user
Private Sub Viewcurrentsubjects_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Viewcurrentsubjects.Click
    'clears the current selection of columns in the viewcurrentvalues datagrid
    Viewcurrentvalues.Columns.Clear()
    'the subject delete state is set to true and all other states are set to false
    subjectdelete = True
    userdelete = False
    Leveldelete = False
    GradeTypeDelete = False
    Yeardelete = False
    useredit = False
    'stores a dataset
    Dim ds As New DataSet

    'sets the SQL statement
    sql = "SELECT DISTINCT Subject.SubjectID, Subject.Subject, Department.Department
FROM Subject, Department WHERE Subject.DepartmentID=Department.DepartmentID ORDER BY
Subject.Subject ASC"
    'the dataset is set after passing the following SQL statement and table into the
GenerateDataset sub of the public module
    ds = generateDataset(sql, "temp")
    'sets the datasource of the datagrid
    Viewcurrentvalues.DataSource = ds
    'sets thedatagrid as visible
    Viewcurrentvalues.Visible = True
    'sets the datamember of thedatagrid
    Viewcurrentvalues.DataMember = "temp"

    'creates a new variable that stores information for a button column that can be
added to thedatagrid
    Dim Columnnedit As New DataGridViewButtonColumn
    With Columnnedit
        'sets the header text of the button column
        .HeaderText = "Record Details"
        'sets the name of the button column
        .Name = "SubjectRecord"
        'sets the text displayed on the buttons of the button column in thedatagrid
        .Text = "Delete Record"
        .UseColumnTextForButtonValue = True
    End With

    'adds the button column to thedatagrid if the number of columns is 3
    If Viewcurrentvalues.Columns.Count = 3 Then
        Viewcurrentvalues.Columns.Add(Columnnedit)
    End If
End Sub
'this sub works like the example above but selects levels instead
Private Sub Viewcurrentlevel_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Viewcurrentlevel.Click
    Viewcurrentvalues.Columns.Clear()
    subjectdelete = False
    userdelete = False
    Leveldelete = True
    GradeTypeDelete = False
    Yeardelete = False
    useredit = False

```

```
Dim ds As New DataSet
sql = "SELECT DISTINCT * FROM Level ORDER BY Level ASC"
ds = generateDataset(sql, "temp")
Viewcurrentvalues.DataSource = ds
Viewcurrentvalues.Visible = True
Viewcurrentvalues.DataMember = "temp"

Dim Columnedit As New DataGridViewButtonColumn
With Columnedit
    .HeaderText = "Record Details"
    .Name = "LevelRecord"
    .Text = "Delete Record"
    .UseColumnTextForButtonValue = True
End With
If Viewcurrentvalues.Columns.Count = 2 Then
    Viewcurrentvalues.Columns.Add(Columnedit)
End If
End Sub
'this sub works like the example above but selects dataset years instead
Private Sub Viewcurrentdatasetyear_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Viewcurrentdatasetyear.Click
    Viewcurrentvalues.Columns.Clear()
    subjectdelete = False
    userdelete = False
    Leveldelete = False
    GradeTypedelete = False
    Yeardelete = True
    useredit = False

    Dim ds As New DataSet
    sql = "SELECT DISTINCT * FROM DatasetID ORDER BY Year ASC"
    ds = generateDataset(sql, "temp")
    Viewcurrentvalues.DataSource = ds
    Viewcurrentvalues.Visible = True
    Viewcurrentvalues.DataMember = "temp"

    Dim Columnedit As New DataGridViewButtonColumn
    With Columnedit
        .HeaderText = "Record Details"
        .Name = "YearRecord"
        .Text = "Delete Record"
        .UseColumnTextForButtonValue = True
    End With
    If Viewcurrentvalues.Columns.Count = 2 Then
        Viewcurrentvalues.Columns.Add(Columnedit)
    End If
End Sub
'this sub works like the example above but selects logins that can be deleted
Private Sub viewlogins_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles viewlogins.Click
    Viewcurrentvalues.Columns.Clear()
    subjectdelete = False
    Leveldelete = False
    userdelete = True
    GradeTypedelete = False
    Yeardelete = False
    useredit = False

    Dim ds As New DataSet
    sql = "SELECT DISTINCT * FROM UserLogin ORDER BY LoginID ASC"
    ds = generateDataset(sql, "temp")
```

```
Viewcurrentvalues.DataSource = ds
Viewcurrentvalues.Visible = True
Viewcurrentvalues.DataMember = "temp"

Dim Columnnedit As New DataGridViewButtonColumn
With Columnnedit
    .HeaderText = "Record Details"
    .Name = "LoginRecord"
    .Text = "Delete Record"
    .UseColumnTextForButtonValue = True
End With
If Viewcurrentvalues.Columns.Count = 4 Then
    Viewcurrentvalues.Columns.Add(Columnnedit)
End If
Viewcurrentvalues.AutoResizeColumns()
End Sub
'this sub works like the example above but selects logins that can be edited
Private Sub vieweditlogins_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles vieweditlogins.Click
    Viewcurrentvalues.Columns.Clear()
    subjectdelete = False
    Leveldelete = False
    userdelete = True
    GradeTypeDelete = False
    YearDelete = False
    useredit = True

    Dim ds As New DataSet
    sql = "SELECT DISTINCT * FROM UserLogin ORDER BY LoginID ASC"
    ds = generateDataset(sql, "temp")
    Viewcurrentvalues.DataSource = ds
    Viewcurrentvalues.Visible = True
    Viewcurrentvalues.DataMember = "temp"

    Dim Columnnedit As New DataGridViewButtonColumn
    With Columnnedit
        .HeaderText = "Record Details"
        .Name = "LoginRecord"
        .Text = "Edit Record"
        .UseColumnTextForButtonValue = True
    End With
    If Viewcurrentvalues.Columns.Count = 4 Then
        Viewcurrentvalues.Columns.Add(Columnnedit)
    End If
    Viewcurrentvalues.AutoResizeColumns()
End Sub
'this sub works like the example above but selects grade types instead
Private Sub Viewcurrentgradetype_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Viewcurrentgradetype.Click
    Viewcurrentvalues.Columns.Clear()
    subjectdelete = False
    Leveldelete = False
    userdelete = False
    GradeTypeDelete = True
    YearDelete = False
    useredit = False

    Dim ds As New DataSet
    sql = "SELECT DISTINCT * FROM GradeType ORDER BY GradeType ASC"
    ds = generateDataset(sql, "temp")
    Viewcurrentvalues.DataSource = ds
    Viewcurrentvalues.Visible = True
```

```

ViewcurrentvaluesDataMember = "temp"

Dim Columnnedit As New DataGridViewButtonColumn
With Columnnedit
    .HeaderText = "Record Details"
    .Name = "GradetypeRecord"
    .Text = "Delete Record"
    .UseColumnTextForButtonValue = True
End With
If Viewcurrentvalues.Columns.Count = 2 Then
    Viewcurrentvalues.Columns.Add(Columnnedit)
End If
End Sub
'this function returns a true or false state depending on whether conditions within
the function have been met or not
Public Function txtvalidatesubject() As Boolean
Call DatabaseConnect()
'initially the functions state is set to true
txtvalidatesubject = True
'if any of the textboxes are left blank then the function returns a false state
and the background of the textbox field changes to red, otherwise the background of the
textbox changes to white
If Newsubjectfield.Text = "" Then
    Newsubjectfield.BackColor = Color.Salmon
    txtvalidatesubject = False
Else
    Newsubjectfield.BackColor = Color.White
End If
If Departmentfield.Text = "" Then
    Departmentfield.BackColor = Color.Salmon
    txtvalidatesubject = False
Else
    Departmentfield.BackColor = Color.White
End If
End Function
'this function works like the one above but deals with the level fields
Public Function txtvalidatelevel() As Boolean
Call DatabaseConnect()
txtvalidatelevel = True
If Levelfield.Text = "" Then
    Levelfield.BackColor = Color.Salmon
    txtvalidatelevel = False
Else
    Levelfield.BackColor = Color.White
End If
End Function
'this function works like the one above but deals with the dataset year fields
Public Function txtvalidatedataset() As Boolean
Call DatabaseConnect()
txtvalidatedataset = True
If Datasetyearfield.Text = "" Then
    Datasetyearfield.BackColor = Color.Salmon
    txtvalidatedataset = False
Else
    Datasetyearfield.BackColor = Color.White
End If
End Function
'this function works like the one above but deals with the gradetype fields
Public Function txtvalidategradetype() As Boolean
Call DatabaseConnect()
txtvalidategradetype = True
If Newgradetypefield.Text = "" Then

```

```

        Newgradetypfield.BackColor = Color.Salmon
        txtvalidategradetype = False
    Else
        Newgradetypfield.BackColor = Color.White
    End If
End Function
'this function works like the one above but deals with the login fields
Public Function txtvalidatelogin() As Boolean
    Call DatabaseConnect()
    txtvalidatelogin = True
    If usernamefield.Text = "" Then
        usernamefield.BackColor = Color.Salmon
        txtvalidatelogin = False
    Else
        usernamefield.BackColor = Color.White
    End If
    If passwordfield.Text = "" Then
        passwordfield.BackColor = Color.Salmon
        txtvalidatelogin = False
    Else
        passwordfield.BackColor = Color.White
    End If
    If adminfield.Text = "" Then
        adminfield.BackColor = Color.Salmon
        txtvalidatelogin = False
    Else
        adminfield.BackColor = Color.White
    End If
End Function
'this sub runs when the user clicks the add subject button, which adds the contents of
the subject fields as a new record into the database
Private Sub Addsubjectbutton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Addsubjectbutton.Click
    'stores a SQL data adapter
    Dim da As New SqlDataAdapter
    'stores a datatable
    Dim dt As New DataTable
    'stores a dataset
    Dim ds As New DataSet
    'stores a studentID
    Dim SID As Integer
    'sets the SQL satatement
    sql = "Select TOP 1 SubjectID FROM Subject ORDER BY SubjectID DESC"
    'stores cmmnd as a SQL command that uses the above SQL statement and the connection
string of con
    Dim cmmnd As New SqlCommand(sql, con)
    'sets the dataadapter select command as cmmnd
    da.SelectCommand = cmmnd
    'the dataadapter retrieves the query results from the SQL statement after
connecting to the Database and fills a dataset
    da.Fill(ds, "Subject")
    'the datatable variable is then set to the table result of the dataset variable
    dt = ds.Tables("Subject")
    'The StudentID is then set as the value of the highest StudentID found from the
Query and then incremented by 1, the SQL statement takes into account removing Students so
that inserting a record will use the first available StudentID
    SID = ds.Tables(0).Rows(0).Item("SubjectID").ToString() + 1

    DatabaseConnect()
    'if the subject related textboxes have all been filled in then the code continues
    If txtvalidatesubject() Then
        'sets the SQL statement

```

```

        sql = "INSERT INTO Subject(SubjectID, DepartmentID, Subject) VALUES
(@SubjectID, @DepartmentID, @Subject)"
    'uses cmd as a new SQL command that uses the above SQL statement and the
connection string of con
    Using cmd = New SqlCommand(sql, con)
        'sets the parameters of the values to be inserted
        cmd.Parameters.AddWithValue("SubjectID", SID)
        cmd.Parameters.AddWithValue("Subject", Newsubjectfield.Text)
        cmd.Parameters.AddWithValue("DepartmentID", Departmentfield.SelectedValue)
        cmd.ExecuteNonQuery()
    End Using
    'closes the database connection
    con.Close()
    'notifies the user with the appropriate message
    MsgBox("Record Inserted", MsgBoxStyle.Information)
    'this commands clicks the viewcurrentsubjects button which essentially
refreshes the datagrid
    Viewcurrentsubjects.PerformClick()
Else
    'if one or more textboxes have not been completed the following error message
displays and nothing happens
    MsgBox("one or more textboxes have not been completed",
MsgBoxStyle.Information, MessageBoxButtons.OK)
End If
End Sub
'this sub runs when the user clicks the add level button, which adds the contents of
the level fields as a new record into the database
'this sub works like the add subject example above
Private Sub Addlevelbutton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Addlevelbutton.Click
    Dim da As New SqlDataAdapter
    Dim dt As New DataTable
    Dim ds As New DataSet
    Dim LID As Integer
    sql = "Select TOP 1 levelID FROM Level ORDER BY LevelID DESC"
    Dim cmmnd As New SqlCommand(sql, con)
    da.SelectCommand = cmmnd
    da.Fill(ds, "Level")
    dt = ds.Tables("Level")
    LID = ds.Tables(0).Rows(0).Item("levelID").ToString() + 1

    DatabaseConnect()
    If txtvalidatelevel() Then
        sql = "INSERT INTO Level(LevelID, Level) VALUES (@LevelID, @Level)"
        Using cmd = New SqlCommand(sql, con)
            cmd.Parameters.AddWithValue("levelID", LID)
            cmd.Parameters.AddWithValue("level", Levelfield.Text)
            cmd.ExecuteNonQuery()
        End Using
        con.Close()
        MsgBox("Record Inserted", MsgBoxStyle.Information)
        Viewcurrentlevel.PerformClick()
    Else
        MsgBox("one or more textboxes have not been completed",
MsgBoxStyle.Information, MessageBoxButtons.OK)
    End If
End Sub
'this sub runs when the user clicks the add dataset button, which adds the contents of
the dataset fields as a new record into the database
'this sub works like the add subject example above
Private Sub Adddatasetyear_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Adddatasetyear.Click

```

```

Dim da As New SqlDataAdapter
Dim dt As New DataTable
Dim ds As New DataSet
Dim DID As Integer
sql = "Select TOP 1 DatasetID FROM DatasetID ORDER BY DatasetID DESC"
Dim cmmnd As New SqlCommand(sql, con)
da.SelectCommand = cmmnd
da.Fill(ds, "DatasetID")
dt = ds.Tables("DatasetID")
DID = ds.Tables(0).Rows(0).Item("DatasetID").ToString() + 1

DatabaseConnect()
If txtvalidatedataset() Then
    sql = "INSERT INTO DatasetID(DatasetID, Year) VALUES (@DatasetID, @Year)"
    Using cmd = New SqlCommand(sql, con)
        cmd.Parameters.AddWithValue("datasetID", DID)
        cmd.Parameters.AddWithValue("year", Datasetyearfield.Text)
        cmd.ExecuteNonQuery()
    End Using
    con.Close()
    MsgBox("Record Inserted", MsgBoxStyle.Information)
    Viewcurrentdatasetyear.PerformClick()
Else
    MsgBox("one or more textboxes have not been completed",
    MsgBoxStyle.Information, MessageBoxButtons.OK)
End If
End Sub
'this sub runs when the user clicks the add gradetype button, which adds the contents
of the gradetype fields as a new record into the database
'this sub works like the add subject example above
Private Sub Addgradatypebutton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Addgradatypebutton.Click
    Dim da As New SqlDataAdapter
    Dim dt As New DataTable
    Dim ds As New DataSet
    Dim GTID As Integer
    sql = "Select TOP 1 GradeTypeID FROM GradeType ORDER BY GradeTypeID DESC"
    Dim cmmnd As New SqlCommand(sql, con)
    da.SelectCommand = cmmnd
    da.Fill(ds, "GradeType")
    dt = ds.Tables("GradeType")
    GTID = ds.Tables(0).Rows(0).Item("GradeTypeID").ToString() + 1

    DatabaseConnect()
    If txtvalidategradetype() Then
        sql = "INSERT INTO GradeType (GradeTypeID, GradeType) VALUES (@GradeTypeID,
@GradeType)"
        Using cmd = New SqlCommand(sql, con)
            cmd.Parameters.AddWithValue("GradeTypeID", GTID)
            cmd.Parameters.AddWithValue("GradeType", Newgradatypefield.Text)
            cmd.ExecuteNonQuery()
        End Using
        con.Close()
        MsgBox("Record Inserted", MsgBoxStyle.Information)
        Viewcurrentgradetype.PerformClick()
    Else
        MsgBox("one or more textboxes have not been completed",
        MsgBoxStyle.Information, MessageBoxButtons.OK)
    End If
End Sub
'this sub runs when the user clicks the add user button, which adds the contents of
the userlogin fields as a new record into the database

```

```

'this sub works like the add subject example above
Private Sub adduser_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles adduser.Click
    Dim da As New SqlDataAdapter
    Dim dt As New DataTable
    Dim ds As New DataSet
    Dim LID As Integer
    sql = "Select TOP 1 LoginID FROM UserLogin ORDER BY LoginID DESC"
    Dim cmmnd As New SqlCommand(sql, con)
    da.SelectCommand = cmmnd
    da.Fill(ds, "UserLogin")
    dt = ds.Tables("UserLogin")
    LID = ds.Tables(0).Rows(0).Item("LoginID").ToString() + 1

    DatabaseConnect()
    If txtvalidateLogin() Then
        sql = "INSERT INTO UserLogin (LoginID, LoginUsername, Password, Admin) VALUES
(@LoginID, @Username, @Password, @Admin)"
        Using cmd = New SqlCommand(sql, con)
            cmd.Parameters.AddWithValue("LoginID", LID)
            cmd.Parameters.AddWithValue("Username", usernamefield.Text)
            cmd.Parameters.AddWithValue("Password", passwordfield.Text)
            cmd.Parameters.AddWithValue("Admin", adminfield.Text)
            cmd.ExecuteNonQuery()
        End Using
        con.Close()
        MsgBox("Record Inserted", MsgBoxStyle.Information)
        viewlogins.PerformClick()
    Else
        MsgBox("one or more textboxes have not been completed",
MsgBoxStyle.Information, MessageBoxButtons.OK)
        End If
    End Sub
    'this sub runs when the admin action field changes text value
    Private Sub Adminaction_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Adminaction.SelectedIndexChanged
        'this case sattement selects a series of cases depending on the user's selection
        choice from the combobox
        Select Case Adminaction.Text
            'if the field is blank then all of these groupboxes are non visible
            Case ""
                Subjectgroup.Visible = False
                Levelgroup.Visible = False
                Addloggingroup.Visible = False
                Viewcurrentvalues.Visible = False
                Gradetypegroup.Visible = False
                Dataset.Visible = False
                Editloggingroup.Visible = False
                'if the field matches this text then all of these groupboxes either become
                visible or non-visible to the user
            Case "Add a new Dataset Year"
                Subjectgroup.Visible = False
                Levelgroup.Visible = False
                Addloggingroup.Visible = False
                Dataset.Visible = True
                Editloggingroup.Visible = False
                Gradetypegroup.Visible = False
                'this programmatically clicks the view dataset year button so that the
                datagrid also loads with the current selection of datasets
                Viewcurrentdatasetyear.PerformClick()
            Case "Add a new Grade Type"
                Subjectgroup.Visible = False

```

```

        Levelgroup.Visible = False
        Addloggingroup.Visible = False
        Dataset.Visible = False
        Editloggingroup.Visible = False
        Gradetypegroup.Visible = True
        Viewcurrentgradetype.PerformClick()
    Case "Add a new Subject"
        Levelgroup.Visible = False
        Addloggingroup.Visible = False
        Gradetypegroup.Visible = False
        Dataset.Visible = False
        Editloggingroup.Visible = False
        Subjectgroup.Visible = True
        Viewcurrentsubjects.PerformClick()
    Case "Add a new Level"
        Subjectgroup.Visible = False
        Addloggingroup.Visible = False
        Gradetypegroup.Visible = False
        Dataset.Visible = False
        Editloggingroup.Visible = False
        Levelgroup.Visible = True
        Viewcurrentlevel.PerformClick()
    Case "Add a new login"
        Subjectgroup.Visible = False
        Gradetypegroup.Visible = False
        Dataset.Visible = False
        Levelgroup.Visible = False
        Addloggingroup.Visible = True
        Editloggingroup.Visible = False
        viewlogins.PerformClick()
    Case "Edit an existing login"
        Subjectgroup.Visible = False
        Gradetypegroup.Visible = False
        Dataset.Visible = False
        Levelgroup.Visible = False
        Addloggingroup.Visible = False
        Editloggingroup.Visible = True
        vieweditlogins.PerformClick()
        Editloggingroup.Visible = False
    End Select
End Sub
'this sub runs when the editlogin button is clicked by the user
Private Sub editlogin_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles editlogin.Click
    Dim da As New SqlDataAdapter
    Dim dt As New DataTable
    Dim ds As New DataSet
    DatabaseConnect()

    'sets the update SQL statement
    sql = "UPDATE UserLogin SET LoginUsername = @LoginUsername, Password = @Password,
Admin = @Admin WHERE LoginID = '" & loginidfield.Text & "'"
    'uses cmd as a new SQL command that uses the above SQL statement and the
connection string of con
    Using cmd = New SqlCommand(sql, con)
        'sets the parameters of the vlues to be updated
        cmd.Parameters.AddWithValue("LoginUsername", usernamefield1.Text)
        cmd.Parameters.AddWithValue("Password", passwordfield1.Text)
        cmd.Parameters.AddWithValue("Admin", adminfield1.Text)
        cmd.ExecuteNonQuery()
    End Using
    'closes the database connection

```

```

        con.Close()
        'notifies the user with the appropriate message
        MsgBox("Record Updated", MsgBoxStyle.Information)
        'programmatically clicks the vieweditlogins sub so that the datagrid refreshes with
the new value that has been updated
        vieweditlogins.PerformClick()
    End Sub
    'processes deleteing a subject
    Sub deletesubject(ByVal e)
        'if the button clicked by the user contains the Delete record text the delete
function continues processing
        If Viewcurrentvalues.Rows(e.RowIndex).Cells("SubjectRecord").Value = "Delete
Record" Then
            'stores varaiables that are set as their corrsponding cell values on the row of
the delete record button clicked by the user
            Dim subjectID = Viewcurrentvalues.Rows(e.RowIndex).Cells("SubjectID").Value
            Dim Subject = Viewcurrentvalues.Rows(e.RowIndex).Cells("Subject").Value
            Dim Department = Viewcurrentvalues.Rows(e.RowIndex).Cells("Department").Value
            DatabaseConnect()
            'displays a messagebox with a yes/no input that asks the user if they really
want to delete the record
            Dim result As Integer = MessageBox.Show("Do you really want to delete this
record?", "Delete", MessageBoxButtons.YesNo, MessageBoxIcon.Warning)
            If result = DialogResult.Yes Then
                'if the result is yes:
                'the SQL delete statement is set
                sql = "DELETE FROM Subject WHERE SubjectID = '" & subjectID & "' AND
Subject = '" & Subject & "'"
                'uses cmd as a new SQL command that uses the above SQL statement and the
connection string of con
                Using cmd = New SqlCommand(sql, con)
                    'executes teh query
                    cmd.ExecuteNonQuery()
                End Using
                con.Close()
                'notifies the user with the following message
                MsgBox("Record Deleted From Database", MsgBoxStyle.Information)
                'this commands clicks the viewcurrentsubjects button which essentially
refreshes thedatagrid
                Viewcurrentsubjects.PerformClick()
                'if the user selects no then the subjectdelete state is set as true and
nothing happens
            ElseIf result = DialogResult.No Then
                subjectdelete = True
            End If
        Else
        End If
    End Sub
    'processes deleteing a dataset
    Sub deletedataset(ByVal e)
        If Viewcurrentvalues.Rows(e.RowIndex).Cells("YearRecord").Value = "Delete Record"
Then
            Dim DatasetID = Viewcurrentvalues.Rows(e.RowIndex).Cells("DatasetID").Value
            Dim Year = Viewcurrentvalues.Rows(e.RowIndex).Cells("Year").Value
            DatabaseConnect()
            Dim result As Integer = MessageBox.Show("Do you really want to delete this
record? THIS CANNOT BE UNDONE", "Delete", MessageBoxButtons.YesNo, MessageBoxIcon.Warning)
            If result = DialogResult.Yes Then
                sql = "DELETE FROM DatasetID WHERE DatasetID = '" & DatasetID & "' AND
Year = '" & Year & "'"
                Using cmd = New SqlCommand(sql, con)
                    cmd.ExecuteNonQuery()

```

```

        End Using
        con.Close()
        MsgBox("Record Deleted From Database", MsgBoxStyle.Information)
        Viewcurrentdatasetyear.PerformClick()
    ElseIf result = DialogResult.No Then
        Yeardelete = True
    End If
Else
End If
End Sub
'processes deleteing a user
Sub deleteuser(ByVal e)
    If Viewcurrentvalues.Rows(e.RowIndex).Cells("LoginRecord").Value = "Delete Record"
Then
    Dim LoginID = Viewcurrentvalues.Rows(e.RowIndex).Cells("LoginID").Value
    Dim LoginUsername =
Viewcurrentvalues.Rows(e.RowIndex).Cells("LoginUsername").Value
    Dim Password = Viewcurrentvalues.Rows(e.RowIndex).Cells("Password").Value
    DatabaseConnect()
    Dim result As Integer = MessageBox.Show("Do you really want to delete this
record?", "Delete", MessageBoxButtons.YesNo, MessageBoxIcon.Warning)
    If result = DialogResult.Yes Then
        sql = "DELETE FROM UserLogin WHERE LoginID = '" & LoginID & "' AND
LoginUsername = '" & LoginUsername & "' AND Password = '" & Password & "'"
        Using cmd = New SqlCommand(sql, con)
            cmd.ExecuteNonQuery()
        End Using
        con.Close()
        MsgBox("Record Deleted From Database", MsgBoxStyle.Information)
        viewlogins.PerformClick()
    ElseIf result = DialogResult.No Then
        userdelete = True
    End If
Else
End If
End Sub
'processes deleteing a gradetype
Sub deletegradetype(ByVal e)
    If Viewcurrentvalues.Rows(e.RowIndex).Cells("GradetypeRecord").Value = "Delete
Record" Then
        Dim GradeTypeID =
Viewcurrentvalues.Rows(e.RowIndex).Cells("GradeTypeID").Value
        Dim GradeType = Viewcurrentvalues.Rows(e.RowIndex).Cells("GradeType").Value
        DatabaseConnect()
        Dim result As Integer = MessageBox.Show("Do you really want to delete this
record?", "Delete", MessageBoxButtons.YesNo, MessageBoxIcon.Warning)
        If result = DialogResult.Yes Then
            sql = "DELETE FROM GradeType WHERE GradeTypeID = '" & GradeTypeID & "' AND
GradeType = '" & GradeType & "'"
            Using cmd = New SqlCommand(sql, con)
                cmd.ExecuteNonQuery()
            End Using
            con.Close()
            MsgBox("Record Deleted From Database", MsgBoxStyle.Information)
            Viewcurrentgradetype.PerformClick()
        ElseIf result = DialogResult.No Then
            GradeTypeDelete = True
        End If
Else
End If
End Sub
'processes editing a user

```

```

Sub edituser(ByVal e)
    If Viewcurrentvalues.Rows(e.RowIndex).Cells("LoginRecord").Value = "Edit Record"
Then

    Dim LoginUsername =
Viewcurrentvalues.Rows(e.RowIndex).Cells("LoginUsername").Value
    Dim Password = Viewcurrentvalues.Rows(e.RowIndex).Cells("Password").Value
    Dim admin = Viewcurrentvalues.Rows(e.RowIndex).Cells("Admin").Value
    Dim loginid = Viewcurrentvalues.Rows(e.RowIndex).Cells("LoginID").Value
    DatabaseConnect()

    usernamefield1.Text = LoginUsername
    passwordfield1.Text = Password
    adminfield1.Text = admin
    loginidfield.Text = loginid
    Editloggingroup.Visible = True
    End If
End Sub
'processes deleteing a level
Sub deletelevel(ByVal e)
    If Viewcurrentvalues.Rows(e.RowIndex).Cells("LevelRecord").Value = "Delete Record"
Then

    Dim LevelID = Viewcurrentvalues.Rows(e.RowIndex).Cells("LevelID").Value
    Dim Level = Viewcurrentvalues.Rows(e.RowIndex).Cells("Level").Value
    DatabaseConnect()
    Dim result As Integer = MessageBox.Show("Do you really want to delete this
record?", "Delete", MessageBoxButtons.YesNo, MessageBoxIcon.Warning)
    If result = DialogResult.Yes Then
        sql = "DELETE FROM Level WHERE LevelID = '" & LevelID & "' AND Level = ''"
& Level & "'"
        Using cmd = New SqlCommand(sql, con)
            cmd.ExecuteNonQuery()
        End Using
        con.Close()
        MsgBox("Record Deleted From Database", MsgBoxStyle.Information)
        Viewcurrentlevel.PerformClick()
    ElseIf result = DialogResult.No Then
        Leveldelete = True
    End If
    Else
    End If
End Sub
'deals with the print form button
Private Sub printbutton_Click(sender As System.Object, e As System.EventArgs) Handles
printbutton.Click
    'sets the printer settings to default
    PrintForm.PrinterSettings = PrintDialog.PrinterSettings
    If PrintDialog.ShowDialog() = DialogResult.OK Then
        'allows the user to change print settings
        PrintForm.PrinterSettings = PrintDialog.PrinterSettings
        'changes the orientation of the printed document to landscape
        Me.PrintForm.PrinterSettings.DefaultPageSettings.Landscape = True
        'refreshes the form so that the printdialog is not printed along with the form
        Me.Refresh()
        'prints the current form
        Me.PrintForm.Print()
        'Percentageschart.Printing.Print(True)
    End If
End Sub

```


Grade Percentages

This form is used to show detailed grade statistics, including predicted grade statistics. A user can use a series of filters to select the statistics for which subject, year, level and grade type they want to view. Two choices of graphs are also available to view these statistics; these include a standard linear graph type and a cumulative graph type. The graph is created dynamically at run time along with the statistics in order to cope with any changes with the overall system. The actual statistics can be viewed in alternate tables view which will show the raw statistics of students achieving or predicted a grade including the number of students achieving or predicted grades A* to C. A hard copy of this form can be produced by clicking the print icon.

Main Menu ▶ Grade Percentages

Grade Percentages

Select a Subject

Select a Subject: English Language

Select a Year: 2014

Select a Level: KS4

Select a Grade Type: A2 Predicted

Select a Graph Type: Standard Grade Percentage Graph

View Percentages

Clear Chart Selection

Change Data View

View Graph

View Number of Students achieving grade

Percentage of Students Predicted a Grade (%)

Grade	Percentage
A*	1.12%
A	8.43%
B	44.38%
C	38.2%
D	7.87%
E	0%
U	0%

Main Menu ▶ Grade Percentages

Grade Percentages

Select a Subject

Select a Subject: English Language

Select a Year: 2014

Select a Level: KS4

Select a Grade Type: A2 Predicted

Select a Graph Type: Standard Grade Percentage Graph

View Percentages

Clear Chart Selection

Change Data View

View Graph

View Number of Students achieving grade

Grade	Number of Student's Predicted Grade
A*	2 out of 178
A	15 out of 178
B	79 out of 178
C	68 out of 178
D	14 out of 178
E	0 out of 178
U	0 out of 178

Number of Students Predicted Grades A*-C

164 out of 178 (~92%)

Addsearch System.Windows.Forms.Button
AtoC System.Windows.Forms.Label
AtoCtable System.Windows.Forms.TableLayoutPanel
banner System.Windows.Forms.PictureBox
chartlabel System.Windows.Forms.Label
Classeslink System.Windows.Forms.LinkLabel
Clearselection System.Windows.Forms.Button
Grade_Percentages System.Windows.Forms.Form
Grade3 System.Windows.Forms.Label
Grade4 System.Windows.Forms.Label
Grade5 System.Windows.Forms.Label
gradeA System.Windows.Forms.Label
gradeAstar System.Windows.Forms.Label
gradeB System.Windows.Forms.Label
gradeC System.Windows.Forms.Label
gradeD System.Windows.Forms.Label
gradeE System.Windows.Forms.Label
gradeF System.Windows.Forms.Label
Gradetypefield System.Windows.Forms.ComboBox
Graphtypefield System.Windows.Forms.ComboBox
Graphview System.Windows.Forms.Button
Groupview System.Windows.Forms.GroupBox
KS2table System.Windows.Forms.TableLayoutPanel
KS4Table System.Windows.Forms.TableLayoutPanel

Levelfield System.Windows.Forms.ComboBox
Levellabel System.Windows.Forms.Label
MainMenu System.Windows.Forms.LinkLabel
Percentageschart System.Windows.Forms.DataVisualization.Charting.Chart
printbutton System.Windows.Forms.Button
PrintDialog System.Windows.Forms.PrintDialog
PrintForm Microsoft.VisualBasic.PowerPacks.Printing.PrintForm
SearchStudent System.Windows.Forms.GroupBox
SubjectField System.Windows.Forms.ComboBox
subjectlabel System.Windows.Forms.Label
table2title System.Windows.Forms.Label
TablesView System.Windows.Forms.Button
tabletitle System.Windows.Forms.Label
Yearfield System.Windows.Forms.ComboBox
yearlabel System.Windows.Forms.Label

Parameter Name	Description
Grade_Percentages	This handles the instance the form loads and is responsible for connecting to the database along with populating any dynamic combo boxes on the form. It also sets the default text value of the graph type combo box.
kS4percentages	Selects all grades within the database and then counts the number of each KS4 grade e.g. A's B's. It then sets the text properties of the values within the statistics table and adds data points to the series of the chart, these data points depend on the user's selection of a standard or cumulative graph type. It then adds the corresponding data point axis labels for each KS4 grade.
KS2percentages	Selects all grades within the database and then counts the number of each KS2 grade e.g. 3's 4's. It then sets the text properties of the values within the statistics table and adds data points to the series of the chart, these data points depend on the user's selection of a standard or cumulative graph type. It then adds the corresponding data point axis labels for each KS2 grade.
textvalidate	This function sets a true or false condition depending on whether any textboxes on the form are blank. It also sets the background colour of textboxes appropriately to indicate to the user that fields have not been completed.

A2Predictions	This sub generates a series of predicted grades by first selecting all students that take that subject, and calculating an average point score for each student based on their GCSE grades. From this average point score a predicted grade is then generated for the selected subject the user want to view predicted grades for. Once all of the predicted grades have been generated, data points are added to the chart series, these are either linear data points or cumulative data points depending on the users graph choice. The label properties of the statistics table are also set as the number of each predicted grade. Corresponding data point axis labels for each predicted grade are also added to the chart.
Addsearch_Click	When the view percentages button is clicked by the user, this sub determines which type of series to add to the chart. It is also responsible for validation and formatting the chart.
MainMenu_LinkClicked	When the main menu link label is clicked all open forms close and the main menu displays.
Graphview_Click	When the view graph button is clicked all objects associated with the graph visible to the user and all objects associated with the statistics table non-visible to the user.
TablesView_Click	When the view table's button is clicked all objects associated with the graph become non-visible to the user and all objects associated with the statistics table become visible to the user.
Clearselection_Click	Creates a new instance of every object on the form (essentially re-loads the form).
printbutton_Click	This sub allows the user to print a hard copy of the form when the print icon is clicked, additional print dialog options are then displayed allowing the user to choose printer and number of copies amongst other settings.
Levelfield_SelectedIndexChanged	Handles the instance the level field is changed, depending on the text value of the level field, the grade type field text property is set and its enabled state is set.

```

Public Class Grade_Percentages
    'stores series as an empty string that can be used within this class
    Dim series As String = Nothing
    'i is used as an index variable in my for next statements
    Dim i As Integer = 0

    'these variables store the number of grades counted
    Dim threegrade As Integer = 0
    Dim fourgrade As Integer = 0
    Dim fivegrade As Integer = 0
    Dim Astargrade As Integer
    Dim Agrade As Integer = 0
    Dim Bgrade As Integer = 0
    Dim Cgrade As Integer = 0
    Dim Dgrade As Integer = 0
    Dim Egrade As Integer = 0
    Dim Fgrade As Integer = 0
    Dim Ggrade As Integer = 0
    'this sub runs when the form initially loads
    Private Sub Grade_Percentages(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Me.Load

        'runs the databaseconnect sub from the public module
        DatabaseConnect()
        'sets the screenarea of the form so that it is displayed center screen
        Me.Left = (Screen.PrimaryScreen.WorkingArea.Width - Me.Width) / 2
        Me.Top = (Screen.PrimaryScreen.WorkingArea.Height - Me.Height) / 2
        'stores dt as datatable
        Dim dt As DataTable

        'Uses the Generate combo function of the module to populate a combobox by passing
        the table, field and needempty variables to the Generatecombo function and then returning
        the result in a datatable
        dt = GenerateCombo("Subject", "SubjectID, Subject", True)
        'sets the datasource of the combobox
        SubjectField.DataSource = dt
        'sets the display member of the combobox
        SubjectField.DisplayMember = "Subject"
        'the index of each item in the combobox is set as their corresponding ID from the
        database
        SubjectField.ValueMember = "SubjectID"

        dt = GenerateCombo("DatasetID", "DatasetID, Year", True)
        Yearfield.DataSource = dt
        Yearfield.DisplayMember = "Year"
        Yearfield.ValueMember = "DatasetID"

        dt = GenerateCombo("Level", "levelID, Level", True)
        Levelfield.DataSource = dt
        Levelfield.DisplayMember = "Level"
        Levelfield.ValueMember = "LevelID"

        dt = GenerateCombo("GradeType", "GradeTypeID, GradeType", True)
        Gradetypefield.DataSource = dt
        Gradetypefield.DisplayMember = "GradeType"
        Gradetypefield.ValueMember = "GradeTypeID"

        'the text of the graphtype combobox is set as following
        Graphtypefield.Text = "Standard Grade Percentage Graph"
    End Sub
End Class

```

```

End Sub
'this sub deals with KS4 percentages
Public Sub KS4percentages()
    'stores ds as a dataset
    Dim ds As New DataSet
    'stores dt as a datatable
    Dim dt As New DataTable

        ' This SQL statement uses a pivot to create a custom set of columns that writes a
        1 or 0 under that column if the corresponding grade belongs under that column, a 0 means
        that row does not have that grade, a 1 in the pivot table means that row does have a grade
        equivalent to its column value
        sql = "SELECT [A*],[A],[B],[C],[D],[E],[F],[G] FROM Grade PIVOT (Count(GradeID)
FOR Grade IN ([A*],[A],[B],[C],[D],[E],[F],[G])) AS PVTable WHERE SubjectID = '' &
SubjectField.SelectedValue & '' and DatasetID = '' & Yearfield.SelectedValue & '' and
LevelID = '' & Levelfield.SelectedValue & '' and GradeTypeID = '' &
Gradetypefield.SelectedValue & ''
        'the dataset is set after passing the following SQL statement and table into the
GenerateDataset sub of the public module
        ds = generateDataset(sql, "temp")
        'the datatable contents is set as the returned dataset once the SQL statement has
filled the dataset
        dt = ds.Tables(0)

        'this for next statement loops itself to the last row containing a grade
        'it counts the number of each grade within the dataset
        For i = 0 To (dt.Rows.Count - 1)
            'if there is a 1 under any of the columns in the dataset for each row then the
            count of that corresponding grade increases by 1

                If dt.Rows(i)("A*").ToString() = "1" Then
                    Astargrade = Astargrade + 1
                End If
                If dt.Rows(i)("A").ToString() = "1" Then
                    Agrade = Agrade + 1
                End If
                If dt.Rows(i)("B").ToString() = "1" Then
                    Bgrade = Bgrade + 1
                End If
                If dt.Rows(i)("C").ToString() = "1" Then
                    Cgrade = Cgrade + 1
                End If
                If dt.Rows(i)("D").ToString() = "1" Then
                    Dgrade = Dgrade + 1
                End If
                If dt.Rows(i)("E").ToString() = "1" Then
                    Egrade = Egrade + 1
                End If
                If dt.Rows(i)("F").ToString() = "1" Then
                    Fgrade = Fgrade + 1
                End If
                If dt.Rows(i)("G").ToString() = "1" Then
                    Ggrade = Ggrade + 1
                End If
            Next

            'the text values of the labels in the table when the tableviewbutton is clicked
            are set to their corresponding number of grades counted from the dataset
            gradeAstar.Text = Astargrade & " out of " & dt.Rows.Count
            gradeA.Text = Agrade & " out of " & dt.Rows.Count
            gradeB.Text = Bgrade & " out of " & dt.Rows.Count
            gradeC.Text = Cgrade & " out of " & dt.Rows.Count

```

```

gradeD.Text = Dgrade & " out of " & dt.Rows.Count
gradeE.Text = Egrade & " out of " & dt.Rows.Count
Utable1.Text = "F"
gradeF.Text = Fgrade & " out of " & dt.Rows.Count
gradeG.Text = Ggrade & " out of " & dt.Rows.Count
KS4Table.Size = New Size(358, 211)

'this sets the atoc label text of the table by adding up all the grades from A* to
c and then dividing by the total number of grades in the dataset, which is then converted
to a percentage and displayed as a percentage
AtoC.Text = (Agrade + Bgrade + Cgrade + Astargrade) & " out of " & dt.Rows.Count &
" (~" & Int(((Agrade + Bgrade + Cgrade + Astargrade) / dt.Rows.Count) * 100) & "%)"

'this if statement selects what type of graph to display based on the user's
selection

If Graphtypefield.Text = "Cumulative Grade Percentage Graph" Then
    'if the user wants to see a cumulative graph then the name of the graph
changes
    series = "Cumulative Grade Percentage"
    'adds a series to the graph
    Percentageschart.Series.Add(series)
    'adds datapoints to the series of the graph, these points work cumulatively so
    that each following data point adds all the previous data points values to it
    Percentageschart.Series(series).Points.AddY(Astargrade / dt.Rows.Count)
    Percentageschart.Series(series).Points.AddY((Agrade + Astargrade) /
    dt.Rows.Count)
    Percentageschart.Series(series).Points.AddY((Bgrade + Astargrade + Agrade) /
    dt.Rows.Count)
    Percentageschart.Series(series).Points.AddY((Cgrade + Astargrade + Agrade +
    Bgrade) / dt.Rows.Count)
    Percentageschart.Series(series).Points.AddY((Dgrade + Astargrade + Agrade +
    Bgrade + Cgrade) / dt.Rows.Count)
    Percentageschart.Series(series).Points.AddY((Egrade + Dgrade + Astargrade +
    Agrade + Bgrade + Cgrade) / dt.Rows.Count)
    Percentageschart.Series(series).Points.AddY((Fgrade + Egrade + Dgrade +
    Astargrade + Agrade + Bgrade + Cgrade) / dt.Rows.Count)
    Percentageschart.Series(series).Points.AddY((Ggrade + Fgrade + Egrade + Dgrade +
    + Astargrade + Agrade + Bgrade + Cgrade) / dt.Rows.Count)
Else
    'if the user wants to see a non-cumulative graph then the name of the graph
changes
    series = "Non-Cumulative Grade Percentage"
    'adds a series to the graph
    Percentageschart.Series.Add(series)
    'adds datapoints to the series of the graph equivalent to the number of grades
counted
    Percentageschart.Series(series).Points.AddY(Astargrade / dt.Rows.Count)
    Percentageschart.Series(series).Points.AddY(Agrade / dt.Rows.Count)
    Percentageschart.Series(series).Points.AddY(Bgrade / dt.Rows.Count)
    Percentageschart.Series(series).Points.AddY(Cgrade / dt.Rows.Count)
    Percentageschart.Series(series).Points.AddY(Dgrade / dt.Rows.Count)
    Percentageschart.Series(series).Points.AddY(Egrade / dt.Rows.Count)
    Percentageschart.Series(series).Points.AddY(Fgrade / dt.Rows.Count)
    Percentageschart.Series(series).Points.AddY(Ggrade / dt.Rows.Count)
End If
'sets the labels of the datapoints on the series
Percentageschart.Series(series).Points(0).AxisLabel = "A*"
Percentageschart.Series(series).Points(1).AxisLabel = "A"
Percentageschart.Series(series).Points(2).AxisLabel = "B"
Percentageschart.Series(series).Points(3).AxisLabel = "C"
Percentageschart.Series(series).Points(4).AxisLabel = "D"

```

```

Percentageschart.Series(series).Points(5).AxisLabel = "E"
Percentageschart.Series(series).Points(6).AxisLabel = "F"
Percentageschart.Series(series).Points(7).AxisLabel = "G"
End Sub
'this sub deals with KS2 percentages and works similarly to the Sub above
Public Sub KS2percentages()
    Dim ds As New DataSet
    Dim dt As New DataTable

        'a pivot table is also used here to return a dataset that writes a 1 or a 0 if
        there is a corrsponding grade on that row under each column
        sql = "SELECT [3],[4],[5] FROM Grade PIVOT (Count(GradeID) FOR Grade IN
        ([3],[4],[5])) AS PVTable WHERE SubjectID = '" & SubjectField.SelectedValue & "' and
        DatasetID = '" & Yearfield.SelectedValue & "' and LevelID = '" & Levelfield.SelectedValue
        & "' and GradeTypeID = '" & Gradetypefield.SelectedValue & "'"
        ds = generateDataset(sql, "temp")
        dt = ds.Tables(0)

        'counts the number of KS2 grades
        For i = 0 To (dt.Rows.Count - 1)
            If dt.Rows(i)("3").ToString() = "1" Then
                threegrade = threegrade + 1
            End If
            If dt.Rows(i)("4").ToString() = "1" Then
                fourgrade = fourgrade + 1
            End If
            If dt.Rows(i)("5").ToString() = "1" Then
                fivegrade = fivegrade + 1
            End If
        Next
        'sets the text values of the labels within the table when the tableview button is
        clicked
        Grade3.Text = threegrade & " out of " & dt.Rows.Count
        Grade4.Text = fourgrade & " out of " & dt.Rows.Count
        Grade5.Text = fivegrade & " out of " & dt.Rows.Count

        'sets the type of series to add to the graph based on whether the user wants to
        see a cumulative or non-cumulative graph
        If Graphtypefield.Text = "Cumulative Grade Percentage Graph" Then
            series = "Cumulative Grade Percentage"
            Percentageschart.Series.Add(series)
            'adds a cumulative series of datapoints
            Percentageschart.Series(series).Points.AddY(((fivegrade) / dt.Rows.Count))
            Percentageschart.Series(series).Points.AddY(((fourgrade + fivegrade) /
            dt.Rows.Count))
            Percentageschart.Series(series).Points.AddY(((threegrade + fourgrade +
            fivegrade) / dt.Rows.Count)))
        Else
            series = "Non-Cumulative Grade Percentage"
            Percentageschart.Series.Add(series)
            'adds a non-cumulative series of datapoints
            Percentageschart.Series(series).Points.AddY(((fivegrade) / dt.Rows.Count))
            Percentageschart.Series(series).Points.AddY(((fourgrade) / dt.Rows.Count))
            Percentageschart.Series(series).Points.AddY(((threegrade) / dt.Rows.Count))
        End If
        'sets the datapoint labels of the graph
        Percentageschart.Series(series).Points(0).AxisLabel = "3"
        Percentageschart.Series(series).Points(1).AxisLabel = "4"
        Percentageschart.Series(series).Points(2).AxisLabel = "5"
    End Sub
    'this function returns a true or false state depending on whether the conditions
    within this sub have been met or not

```

```

Public Function textvalidate() As Boolean
    'initially the functions state is set to true
    textvalidate = True

    'if any of the textboxes are left blank then the function returns a false state
    and the background of the textbox field changes to red, otherwise the background of the
    textbox changes to white
    If Levelfield.Text = "" Then
        textvalidate = False
        Levelfield.BackColor = Color.Salmon
    Else
        Levelfield.BackColor = Color.White
    End If
    If SubjectField.Text = "" Then
        textvalidate = False
        SubjectField.BackColor = Color.Salmon
    Else
        SubjectField.BackColor = Color.White
    End If
    If Gradetypefield.Text = "" Then
        textvalidate = False
        Gradetypefield.BackColor = Color.Salmon
    Else
        Gradetypefield.BackColor = Color.White
    End If
    If Yearfield.Text = "" Then
        textvalidate = False
        Yearfield.BackColor = Color.Salmon
    Else
        Yearfield.BackColor = Color.White
    End If

End Function
Public Sub A2Predictions(ByVal sql2 As String)
    'stores subjects
    Dim dt As New DataTable
    'stores students
    Dim dt1 As New DataTable
    'stores studentsgrades
    Dim dt2 As New DataTable
    'stores the grid that will be displayed
    Dim dt3 As New DataTable
    Dim ds As DataSet
    'stores index variables used in my for next statements
    Dim i As Integer
    Dim s As Integer

    'stores the predictedvalue point score of a Grade
    Dim predictedvalue As Double
    'stores the total of all the predictedvalues
    Dim currentpredictedtotal As Double = 0
    'stores the actual predicted grade
    Dim predictedgrade As String
    'stores a subject
    Dim subject As String
    'stores a weighting variable that is applied to the predictedvalue depending on
    the subject
    Dim GCSEgradeweighting As Double
    'stores the averagepointscore for a student
    Dim averagepointscore As Double
    'stores the ID of a student
    Dim StudentID As Integer

```

```

'queries all of the subjects and places them in alphabetical order
sql = "SELECT DISTINCT Subject FROM Subject Where Subject.Subject = '' &
SubjectField.Text & '' Order By Subject ASC"
'the dataset is set after passing the following SQL statement and table into the
GenerateDataset sub of the public module
ds = generateDataset(sql, "temp")
'the dt datatable is filled with the a list of subjects from the database
dt = ds.Tables(0)

'the dataset is set after passing the SQL2 statement which is set depending on the
user's filter selection, and table into the GenerateDataset sub of the public module
ds = generateDataset(sql2, "temp")
'the dt1 table is filled with a list of students
dt1 = ds.Tables(0)

'this for next loop loops through every student
'for eachs student the dt2 datatable is filled with that students grades
dt3.Columns.Add("Grade")
For i = 0 To (dt1.Rows.Count - 1)
    averagepointscore = 0
    currentpredictedtotal = 0
    StudentID = dt1.Rows(i)("StudentID").ToString()
    sql = "SELECT DISTINCT Grade.GradeID, Student.Firstname, Student.Surname,
Subject.Subject, Grade.Grade, [Level].[Level], GradeType.GradeType, DatasetID.Year FROM
StudentGroup INNER JOIN Class ON StudentGroup.ClassID = Class.ClassID INNER JOIN Grade
INNER JOIN[Level] ON Grade.LevelID = [Level].LevelID INNER JOIN GradeType ON
Grade.GradeTypeID = GradeType.GradeTypeID INNER JOIN Student ON Grade.StudentID =
Student.StudentID ON StudentGroup.StudentID = Student.StudentID INNER JOIN Subject ON
Grade.SubjectID = Subject.SubjectID INNER JOIN DatasetID ON Grade.DatasetID =
DatasetID.DatasetID WHERE Student.StudentId = '' & StudentID & '' ORDER BY GradeID ASC"
    ds = generateDataset(sql, "temp")
    dt2 = ds.Tables(0)

'this variable is used to count the number of GCSE grades a student has
Dim count As Integer
count = 0

'this for next statement then loops through all of a students grades and then
applies a predictedvalue depending on the grade, this grade later has a weighting applied
to it
For s = 0 To (dt2.Rows.Count - 1)

    'if the value of the grade underneath the Grade header in the datatable on
    the row of the i index is an A* then this if statement runs
    If dt2.Rows(s)("Grade").ToString() = "A*" Then
        'the GCSE gradeweighting variable is recieved after passing the
        datatable and the row index into the GCSEgradeweighting function in the public module
        GCSEgradeweighting = GCSEGradeweighting(s, dt2)
        'the predicted value is set as a national value but then is multiplied
        by the weighting value
        predictedvalue = 58 * GCSEgradeweighting
        count = count + 1

    End If
    'all of the following if statements work like the one above for all the
    other different grades
    If dt2.Rows(s)("Grade").ToString() = "A" Then
        GCSEgradeweighting = GCSEGradeweighting(s, dt2)
        predictedvalue = 52 * GCSEgradeweighting
        count = count + 1

```

```

End If
If dt2.Rows(s)("Grade").ToString() = "B" Then
    GCSEgradeweighting = GCSEGradeweighting(s, dt2)
    predictedvalue = 46 * GCSEgradeweighting
    count = count + 1

End If
If dt2.Rows(s)("Grade").ToString() = "C" Then
    GCSEgradeweighting = GCSEGradeweighting(s, dt2)
    predictedvalue = 40 * GCSEgradeweighting
    count = count + 1

End If
If dt2.Rows(s)("Grade").ToString() = "D" Then
    GCSEgradeweighting = GCSEGradeweighting(s, dt2)
    predictedvalue = 34 * GCSEgradeweighting
    count = count + 1

End If
If dt2.Rows(s)("Grade").ToString() = "E" Then
    GCSEgradeweighting = GCSEGradeweighting(s, dt2)
    predictedvalue = 28 * GCSEgradeweighting
    count = count + 1

End If
If dt2.Rows(s)("Grade").ToString() = "F" Then
    GCSEgradeweighting = GCSEGradeweighting(s, dt2)
    predictedvalue = 22 * GCSEgradeweighting
    count = count + 1
End If
If dt2.Rows(s)("Grade").ToString() = "G" Then
    GCSEgradeweighting = GCSEGradeweighting(s, dt2)
    predictedvalue = 16 * GCSEgradeweighting
    count = count + 1
End If
If dt2.Rows(s)("Grade").ToString() = "4" Then
    GCSEgradeweighting = GCSEGradeweighting(s, dt2)
    predictedvalue = 0
End If
If dt2.Rows(s)("Grade").ToString() = "5" Then
    GCSEgradeweighting = GCSEGradeweighting(s, dt2)
    predictedvalue = 0
End If
'the current predicted total equals the previous currentpredicted total +
the predicted value
currentpredictedtotal = currentpredictedtotal + predictedvalue
Next

'a mean average point score for the student is created by taking the total of
all the predicted values and dividing by the number of grades
averagepointscore = currentpredictedtotal / count

'the subject variable is set as the value of the subject field
subject = SubjectField.Text
'the actual predicted grade is set by passing the subject, row index,
datatable and average point score to the A2predictedgrade sub in the public module
predictedgrade = A2predictedgrade(subject, 0, dt, averagepointscore)
'the predicted grade is then written on the row index of i underneath the
grade header
dt3.Rows.Add()
dt3.Rows(i)("Grade") = predictedgrade
Next

```

```

'this for next statement loops itself to the last row containing a grade
'it counts the number of each grade within the dataset
For i = 0 To (dt3.Rows.Count - 1)
    'if there is a 1 under any of the columns in the dataset for each row then the
    count of that corresponding grade increases by 1

        If dt3.Rows(i)("Grade").ToString() = "A*" Then
            Astargrade = Astargrade + 1
        End If
        If dt3.Rows(i)("Grade").ToString() = "A" Then
            Agrade = Agrade + 1
        End If
        If dt3.Rows(i)("Grade").ToString() = "B" Then
            Bgrade = Bgrade + 1
        End If
        If dt3.Rows(i)("Grade").ToString() = "C" Then
            Cgrade = Cgrade + 1
        End If
        If dt3.Rows(i)("Grade").ToString() = "D" Then
            Dgrade = Dgrade + 1
        End If
        If dt3.Rows(i)("Grade").ToString() = "E" Then
            Egrade = Egrade + 1
        End If
        If dt3.Rows(i)("Grade").ToString() = "F" Then
            Fgrade = Fgrade + 1
        End If
    Next

'the text values of the labels in the table when the tableviewbutton is clicked
are set to their corresponding number of grades counted from the dataset
gradeAstar.Text = Astargrade & " out of " & dt3.Rows.Count
gradeA.Text = Agrade & " out of " & dt3.Rows.Count
gradeB.Text = Bgrade & " out of " & dt3.Rows.Count
gradeC.Text = Cgrade & " out of " & dt3.Rows.Count
gradeD.Text = Dgrade & " out of " & dt3.Rows.Count
gradeE.Text = Egrade & " out of " & dt3.Rows.Count
Utable1.Text = "U"
gradeF.Text = Fgrade & " out of " & dt3.Rows.Count
KS4Table.Size = New Size(358, 187)

'this sets the atoc label text of the table by adding up all the grades from A* to
c and then dividing by the total number of grades in the dataset, which is then converted
to a percentage and displayed as a percentage
Atoc.Text = (Agrade + Bgrade + Cgrade + Astargrade) & " out of " & dt3.Rows.Count
& " (~" & Int(((Agrade + Bgrade + Cgrade + Astargrade) / dt3.Rows.Count) * 100) & "%)"

'this if statement selects what type of graph to display based on the user's
selection

If Graphtypefield.Text = "Cumulative Grade Percentage Graph" Then
    'if the user wants to see a cumulative graph then the name of the graph
changes
    series = "Cumulative Grade Percentage"
    'adds a series to the graph
    Percentageschart.Series.Add(series)
    'adds datapoints to the series of the graph, these points work cumulatively so
    that each following data point adds all the previous data points values to it
    Percentageschart.Series(series).Points.AddY(Astargrade / dt3.Rows.Count)
    Percentageschart.Series(series).Points.AddY((Agrade + Astargrade) /
    dt3.Rows.Count)

```

```

    Percentageschart.Series(series).Points.AddY((Bgrade + Astargrade + Agrade) / 
dt3.Rows.Count)
    Percentageschart.Series(series).Points.AddY((Cgrade + Astargrade + Agrade + 
Bgrade) / dt3.Rows.Count)
    Percentageschart.Series(series).Points.AddY((Dgrade + Astargrade + Agrade + 
Bgrade + Cgrade) / dt3.Rows.Count)
    Percentageschart.Series(series).Points.AddY((Egrade + Dgrade + Astargrade + 
Agrade + Bgrade + Cgrade) / dt3.Rows.Count)
    Percentageschart.Series(series).Points.AddY((Fgrade + Egrade + Dgrade + 
Astargrade + Agrade + Bgrade + Cgrade) / dt3.Rows.Count)
Else
    'if the user wants to see a non-cumulative graph then the name of the graph
changes
    series = "Non-Cumulative Grade Percentage"
    'adds a series to the graph
    Percentageschart.Series.Add(series)
    'adds datapoints to the series of the graph equivalent to the number of grades
counted
    Percentageschart.Series(series).Points.AddY(Astargrade / dt3.Rows.Count)
    Percentageschart.Series(series).Points.AddY(Agrade / dt3.Rows.Count)
    Percentageschart.Series(series).Points.AddY(Bgrade / dt3.Rows.Count)
    Percentageschart.Series(series).Points.AddY(Cgrade / dt3.Rows.Count)
    Percentageschart.Series(series).Points.AddY(Dgrade / dt3.Rows.Count)
    Percentageschart.Series(series).Points.AddY(Egrade / dt3.Rows.Count)
    Percentageschart.Series(series).Points.AddY(Fgrade / dt3.Rows.Count)
End If
'sets the labels of the datapoints on the series
Percentageschart.Series(series).Points(0).AxisLabel = "A*"
Percentageschart.Series(series).Points(1).AxisLabel = "A"
Percentageschart.Series(series).Points(2).AxisLabel = "B"
Percentageschart.Series(series).Points(3).AxisLabel = "C"
Percentageschart.Series(series).Points(4).AxisLabel = "D"
Percentageschart.Series(series).Points(5).AxisLabel = "E"
Percentageschart.Series(series).Points(6).AxisLabel = "U"
End Sub
'this sub runs when the view percentages button is clicked by the user
Private Sub Addsearch_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Addsearch.Click
    'if the state of the txtvalidate function is true then the following code runs
    If textvalidate() = True Then
        'this if statement validates the users selection, since only english language
or maths are stored for KS2, the program prevents the user from displaying an empty graph
for KS2 if any other subject is chosen. An appropriate error message is also displayed
        If Levelfield.Text = "KS2" And SubjectField.Text <> "English Language" And
SubjectField.Text <> "Maths" Then
            MsgBox("That subject is not a part of that KeyStage(Level)",
MsgBoxStyle.Information, MessageBoxButtons.OK)
        Else
            'the values of the following variables are all reset to 0 if they have
been previously set
            Astargrade = 0
            Agrade = 0
            Bgrade = 0
            Cgrade = 0
            Dgrade = 0
            Egrade = 0
            Fgrade = 0
            Ggrade = 0
            threegrade = 0
            fourgrade = 0
            fivegrade = 0
            i = 0

```

```

'calls the Loadgraphic class
Dim objPlsWait As New clsOPAWaitScreen
objPlsWait.ShowWaitScreen("")

'the current graph series and title labels are cleared if they have
already been set
Percentageschart.Series.Clear()
Percentageschart.Titles.Clear()

If Gradetypefield.Text = "A2 Predicted" Then

    'selects all students within the year the user has selected in the
yearfield
    Dim sql2 As String
    sql2 = "SELECT DISTINCT Student.StudentID, Student.Firstname,
Student.Surname FROM StudentGroup INNER JOIN Class ON StudentGroup.ClassID = Class.ClassID
INNER JOIN Grade INNER JOIN[Level] ON Grade.LevelID = [Level].LevelID INNER JOIN GradeType
ON Grade.GradeTypeID = GradeType.GradeTypeID INNER JOIN Student ON Grade.StudentID =
Student.StudentID ON StudentGroup.StudentID = Student.StudentID INNER JOIN Subject ON
Grade.SubjectID = Subject.SubjectID INNER JOIN DatasetID ON Grade.DatasetID =
DatasetID.DatasetID WHERE Student.DatasetID = '" & Yearfield.SelectedValue & "' ORDER BY
Student.StudentID ASC"
    'passes the SQL statement to the A2predictions sub
A2Predictions(sql2)

'sets the chart label text depending on the users graphtype selection
If Graphtypefield.Text = "Cumulative Grade Percentage Graph" Then
    chartlabel.Text = "Cumulative Predictions by Grade (%)"
Else
    chartlabel.Text = "Percentage of Students Predicted a Grade (%)"
End If

'sets the table labels text depending on the users graphtype selection
If Gradetypefield.Text = "A2 Predicted" Then
    tablettitle.Text = "Number of Student's Predicted Grade"
    table2title.Text = "Number of Students Predicted Grades A*-C"
Else
    tablettitle.Text = "Number of Student's Achieving Grade"
    table2title.Text = "Number of Students Achieved Grades A*-C"
End If

'sets which objects on the form are visible
AtoCtable.Visible = True
KS4Table.Visible = True
KS2table.Visible = False
chartlabel.Visible = True
Percentageschart.Visible = True

ElseIf Gradetypefield.Text <> "A2 Predicted" Then

    'sets the chart label text depending on the users graphtype selection
    If Graphtypefield.Text = "Cumulative Grade Percentage Graph" Then
        chartlabel.Text = "Cumulative Percentage by Grade (%)"
    Else
        chartlabel.Text = "Percentage of Students Achieving a Grade (%)"
    End If

    'sets the table labels text depending on the users graphtype selection
    If Gradetypefield.Text <> "A2 Predicted" Then
        tablettitle.Text = "Number of Student's Achieving Grade"

```

```

        table2title.Text = "Number of Students Achieved Grades A*-C"
    Else
        tablettitle.Text = "Number of Student's Predicted Grade"
        table2title.Text = "Number of Students Predicted Grades A*-C"
    End If

    If Levelfield.Text = "KS4" Then
        'if the levelfield value is KS4 then this code runs
        'runs the KS4 percentages sub
        KS4percentages()
        'sets which objects on the form are visible
        AtoCtable.Visible = True
        KS4Table.Visible = True
        KS2table.Visible = False
        chartlabel.Visible = True
        Percentageschart.Visible = True
    Else
        'if the levelfield value is not KS4 then this code runs
        'runs the KS" percentages sub
        KS2percentages()
        'sets which objects on the form are visible
        AtoCtable.Visible = False
        KS4Table.Visible = False
        KS2table.Visible = True
        chartlabel.Visible = True
        Percentageschart.Visible = True
    End If
End If

'disables the legend from the graph
Percentageschart.Series(0).IsVisibleInLegend = False
'shows the values of datapoints
Percentageschart.Series(0).IsValueShownAsLabel = True
'sets the colour of the datapoint labels as maroon
Percentageschart.Series(0).LabelForeColor = Color.Maroon
'changes the chart colour to skyblue
Percentageschart.Series(0).Color = Color.SkyBlue
'changes the chart to a 3D bar style
Percentageschart.ChartAreas(0).Area3DStyle.Enable3D = True
'sets the minimum and maximum axis
Percentageschart.ChartAreas(0).AxisY.Minimum = 0
Percentageschart.ChartAreas(0).AxisY.Maximum = 1
'sets the format of the chart labels
Percentageschart.ChartAreas(0).AxisY.LabelStyle.Format = "#%"
'sets the format of the series labels
Percentageschart.Series(0).LabelFormat = "##.##" & "%"
'sets the colour of the axis labels
Percentageschart.ChartAreas(0).AxisX.LabelStyle.ForeColor = Color.Maroon
Percentageschart.ChartAreas(0).AxisY.LabelStyle.ForeColor = Color.Maroon
'sets the background of the chart as transparent
Percentageschart.ChartAreas(0).BackColor = Color.Transparent

'the following object is set to visible to the user
Groupview.Visible = True
'closes the loadgraphic wait and resets its objplswait variable
objPlsWait.CloseWaitScreen()
objPlsWait = Nothing
End If
Else
    'if the state of the txtvalidate function is not true then the following error
    message displays and nothing else happens

```

```
    MsgBox("One or more textboxes have not been completed",
MsgBoxStyle.Information, MessageBoxButtons.OK)
End If
'throbber.Visible = False
End Sub
'this sub runs when the main menu is clicked by the user
Private Sub MainMenu_LinkClicked(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles MainMenu.LinkClicked
'shows or closes the following forms
Main_Menu.Show()
Me.Close()
End Sub
'this sub runs when the view graph button is clicked by the user
Private Sub Graphview_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Graphview.Click
'the following objects are set to either visible or non-visible
'the table and table headings are no longer visible
Percentageschart.Visible = True
KS4Table.Visible = False
KS2table.Visible = False
AtoCtable.Visible = False
chartlabel.Visible = True
End Sub
'this sub runs when the view number of students... button is clicked by the user
Private Sub TablesView_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles TablesView.Click
'the following objects are set to either visible or non-visible
Percentageschart.Visible = False
chartlabel.Visible = False
'sets which tables to display depending on the level the usr has selected to view
If Levelfield.Text = "KS4" Then
    AtoCtable.Visible = True
    KS4Table.Visible = True
    KS2table.Visible = False
Else
    KS2table.Visible = True
    AtoCtable.Visible = False
    KS4Table.Visible = False
End If
End Sub
'this sub is run when the Clear Selection button is clicked
Private Sub Clearselection_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Clearselection.Click
'clears all controls on this form
Me.Controls.Clear()
're-initializes all components of this form
InitializeComponent()
'then re-loads this form with all components reset
Grade_Percentages(e, e)
End Sub

'deals with the print form button
Private Sub printbutton_Click(sender As System.Object, e As System.EventArgs) Handles
printbutton.Click
'sets the printer settings to default
PrintForm.PrinterSettings = PrintDialog.PrinterSettings
If PrintDialog.ShowDialog() = DialogResult.OK Then
    'allows the user to change print settings
    PrintForm.PrinterSettings = PrintDialog.PrinterSettings
    'changes the orientation of the printed document to landscape
    Me.PrintForm.PrinterSettings.DefaultPageSettings.Landscape = True
    'refreshes the form so that the printdialog is not printed along with the form
```

```
Me.Refresh()
'prints the current form
Me.PrintForm.Print()
'Percentageschart.Printing.Print(True)
End If
End Sub

Private Sub Levelfield_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Levelfield.SelectedIndexChanged
If Levelfield.Text = "KS2" Then
    Gradetypefield.Enabled = False
    Gradetypefield.Text = "Achieved"
Else
    Gradetypefield.Enabled = True
End If
End Sub
End Class
```

Grade

This form allows a user to filter a selection of grades from the database using a series of filter boxes which can be used in conjunction with one another. The filter selection can be easily reset by using the clear selection button. Once a selection of grades has been displayed a user can then edit a particular grade by clicking the corresponding edit grade button of the row the user wants to delete. From this form a new grade can be added to the system for a student and the user can go to an alternative form where they can solely view all grades that a particular student has achieved and the subject details associated with that grade. A hard copy of this form can also be produced if the user needs evidence of search criteria, which can be done by clicking the print icon.

The screenshot shows a Windows application titled "Grade". The main window has a title bar with "Grade" and standard window controls. Below the title bar is a navigation bar with "Main Menu" and "Grade". The main area is titled "Grade" and contains a sub-header "View/Edit a students grade". On the left, there are five filter input fields: "Firstname", "Surname", "Subject", "Grade", and "Level", each with a dropdown arrow. Below these are three buttons: "Search", "ClearSelection", and a blue button labeled "Add a Grade or view a Student's Grades". To the right is a "DataGridView" displaying a list of student grades. The columns are: Firstname, Surname, Subject, Grade, Level, GradeType, Year, and Grade Details. Each row in the grid has an "Edit Grade" button in the last column. The data in the grid is as follows:

Firstname	Surname	Subject	Grade	Level	GradeType	Year	Grade Details
Abdullah	YATES	Computing	E	KS4	Achieved	2013	Edit Grade
Harry	LEWIS	Geography	D	KS4	Achieved	2013	Edit Grade
Anna	WARD	Geography	D	KS4	Achieved	2013	Edit Grade
Harry	LEWIS	History	D	KS4	Achieved	2013	Edit Grade
Anna	WARD	History	D	KS4	Achieved	2013	Edit Grade
Catherine	SUTCLIFFE	Health And Social care	D	KS4	Achieved	2013	Edit Grade
Toby	BAIN	Leisure and Tourism	D	KS4	Achieved	2013	Edit Grade
Sophie	BIRKS	Business Studies	D	KS4	Achieved	2013	Edit Grade
Jed	HORTON	Business Studies	D	KS4	Achieved	2013	Edit Grade
Jasmine	HOWE	Business Studies	D	KS4	Achieved	2013	Edit Grade
Robert	BYWATER	Physical Education	D	KS4	Achieved	2013	Edit Grade
Edward	RAYBOULD	Resistant Materials	D	KS4	Achieved	2013	Edit Grade
Leo	GILLER	Textiles	D	KS4	Achieved	2013	Edit Grade
Connor	SIMMS	Computing	D	KS4	Achieved	2013	Edit Grade
Alice	WARD	Computing	D	KS4	Achieved	2013	Edit Grade

```

Addbutton System.Windows.Forms.Button
banner System.Windows.Forms.PictureBox
Clearselection System.Windows.Forms.Button
Firstnamefield System.Windows.Forms.TextBox
FirnameLabel System.Windows.Forms.Label
Formheader System.Windows.Forms.Label
Grade System.Windows.Forms.Form
Gradefield System.Windows.Forms.ComboBox
Gradegrid System.Windows.Forms.DataGridView
Gradelabel System.Windows.Forms.Label
Gradelink System.Windows.Forms.LinkLabel
Levelfield System.Windows.Forms.ComboBox
Levellabel System.Windows.Forms.Label
linkarrow System.Windows.Forms.Label
MainMenu System.Windows.Forms.LinkLabel
printbutton System.Windows.Forms.Button
PrintDialog System.Windows.Forms.PrintDialog
PrintForm Microsoft.VisualBasic.PowerPacks.Printing.PrintForm
Search System.Windows.Forms.Button
Subjectfield System.Windows.Forms.ComboBox
Subjectgrid System.Windows.Forms.DataGridView
Subjectlabel System.Windows.Forms.Label
SurnameField System.Windows.Forms.TextBox
Surnamelabel System.Windows.Forms.Label
viewstudentgroup System.Windows.Forms.GroupBox
Yearfield System.Windows.Forms.ComboBox
Yearlabel System.Windows.Forms.Label

```

Parameter Name	Description
Grade_Load	This handles the instance the form loads and is responsible for connecting to the database along with populating any dynamic combo boxes on the form. It also sets the default text value of the graph type combo box.
Search_Click	This sub runs when the Search button is clicked and returns a query matching the criteria the user has used to filter their Grade selection by, this selection is then displayed in the data grid. This filter selection is unique because several filters can be used in conjunction with one another.
Subjectgrid_CellClick	When one of the edit grade buttons that is a part of the data grid is clicked by the user the corresponding Grade ID of the record that is on the same row as the button clicked is passed to the edit grade form and the edit grade form displays to the user.
MainMenu_LinkClicked	When the main menu link label is clicked all open forms close and the main menu displays.
Clearselection_Click	Creates a new instance of every object on the form (essentially re-loads the form).
Addbutton_Click	When this button is clicked, this form closes and the Add grade form displays.
printbutton_Click	This sub allows the user to print a hard copy of the form when the print icon is clicked, additional print dialog options are then displayed allowing the user to choose printer and number of copies amongst other settings.

```

'Imports the following system code commands so that I can use them appropriately
Imports System.Data.SqlClient
Imports System.Data
Imports System.Configuration
Imports System.Data.OleDb

Public Class Grade
    'runs the following sub when the form initially loads
    Private Sub Grade_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Me.Load
    'Calls the database connect sub from the module
    DatabaseConnect()

    'sets the screen dimensions to automatically fit centerscreen
    Me.Left = (Screen.PrimaryScreen.WorkingArea.Width - Me.Width) / 2
    Me.Top = (Screen.PrimaryScreen.WorkingArea.Height - Me.Height) / 2
    'clears the current datagrid column selection
    Gradegrid.Columns.Clear()
    'stores dt as a datatable
    Dim dt As New DataTable

    'Uses the Generate combo function of the module to populate a combobox by passing
    the table, field and needempty variables to the Generatecombo function and then returning
    the result in a datatable
    dt = GenerateCombo("Subject", "Subject", True)
    'sets the datasource of the combobox
    Subjectfield.DataSource = dt
    'sets the display member of the combobox
    Subjectfield.DisplayMember = "Subject"

    dt = GenerateCombo("Grade", "Grade", True)
    Gradefield.DataSource = dt
    Gradefield.DisplayMember = "Grade"

    dt = GenerateCombo("Level", "Level", True)
    Levelfield.DataSource = dt
    Levelfield.DisplayMember = "Level"
End Sub
'this sub runs when the search button is clicked
Private Sub Search_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Search.Click
    'clears the current subject grid column selection
    Subjectgrid.Columns.Clear()
    Dim ds As New DataSet
    'stores a number of SQL strings
    Dim sql1 As String
    Dim sql2 As String
    Dim sql3 As String
    Dim sql5 As String
    Dim sql6 As String
    Dim sql7 As String

    'calls the Loadgraphic class
    Dim objPlsWait As New clsOPAWaitScreen
    objPlsWait.ShowWaitScreen("")

    'If either of the selected fields on the form are not blank then these SQL
    statements are set as the following which later form a larger SQL WHERE command
    If Subjectfield.Text <> "" Then
        sql1 = " and Subject.Subject = '" & Subjectfield.Text & "'"
    Else

```

```

        sql1 = ""
End If
If Gradefield.Text <> "" Then
    sql2 = " and Grade.Grade = '" & Gradefield.Text & "'"
Else
    sql2 = ""
End If
If Levelfield.Text <> "" Then
    sql3 = " and Level.Level = '" & Levelfield.Text & "'"
Else
    sql3 = ""
End If
If Firstnamefield.Text <> "" Then
    sql5 = " and Student.Firstname LIKE '%" & Firstnamefield.Text & "%'"
Else
    sql5 = ""
End If
If SurnameField.Text <> "" Then
    sql6 = " and Student.Surname LIKE '%" & SurnameField.Text & "%'"
Else
    sql6 = ""
End If
sql7 = " ORDER BY GradeID ASC"

'sets the SQL statement, this statement is formed by concatenating all of the
above SQL strings to this string to form a dynamic SQL string that changes depending on
the filters the user wants to use in order to query the Grade selection
sql = "SELECT DISTINCT Grade.GradeID, Student.Firstname, Student.Surname,
Subject.Subject, Grade.Grade, [Level].[Level], GradeType.GradeType, DatasetID.Year FROM
StudentGroup INNER JOIN Class ON StudentGroup.ClassID = Class.ClassID INNER JOIN Grade
INNER JOIN[Level] ON Grade.LevelID = [Level].LevelID INNER JOIN GradeType ON
Grade.GradeTypeID = GradeType.GradeTypeID INNER JOIN Student ON Grade.StudentID =
Student.StudentID ON StudentGroup.StudentID = Student.StudentID INNER JOIN Subject ON
Grade.SubjectID = Subject.SubjectID INNER JOIN DatasetID ON Grade.DatasetID =
DatasetID.DatasetID WHERE DatasetID.DatasetID = '" & moduleDataset & "' & sql1 & sql2 &
sql3 & sql5 & sql6 & sql7
'the dataset is set after passing the following SQL statement and table into the
GenerateDataset sub of the public module
ds = generateDataset(sql, "temp")
'sets the datasource of the datagrid
Subjectgrid.DataSource = ds
'sets the datagrid as visible
Subjectgrid.Visible = True
'sets the datamember of the datagrid
Subjectgrid.DataMember = "temp"
'hides the GradeID column from the user
Subjectgrid.Columns("GradeID").Visible = False

'creates a new variable that stores information for a button column that can be
added to the datagrid
Dim Column As New DataGridViewButtonColumn
With Column
    'sets the header text of the button column
    .HeaderText = "Grade Details"
    'sets the name of the button column
    .Name = "Details"
    'sets the text displayed on the buttons of the button column in the datagrid
    .Text = "Edit Grade"
    .UseColumnTextForButtonValue = True
End With
'adds the button column to the datagrid
Subjectgrid.Columns.Add(Column)

```

```

'auto resizes the column widths of the datagrid
Subjectgrid.AutoResizeColumns()

'closes the Loadgraphic form and resets its objplswait variable
objPlsWait.CloseWaitScreen()
objPlsWait = Nothing

End Sub
'if a button that is part of thedatagrid is clicked then the following sub is run
Private Sub Subjectgrid_CellClick(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.DataGridViewCellEventArgs) Handles Subjectgrid.CellClick
    'stores a GradeID
    Dim GradeID As String
    If e.ColumnIndex = 8 Then
        Edit_Grade.Close()
        'the GradeID variable is set as teh value of the cell that is on the same row
        'as the button that is clicked and is under the GradeID header column
        GradeID = Subjectgrid.Rows(e.RowIndex).Cells("GradeID").Value

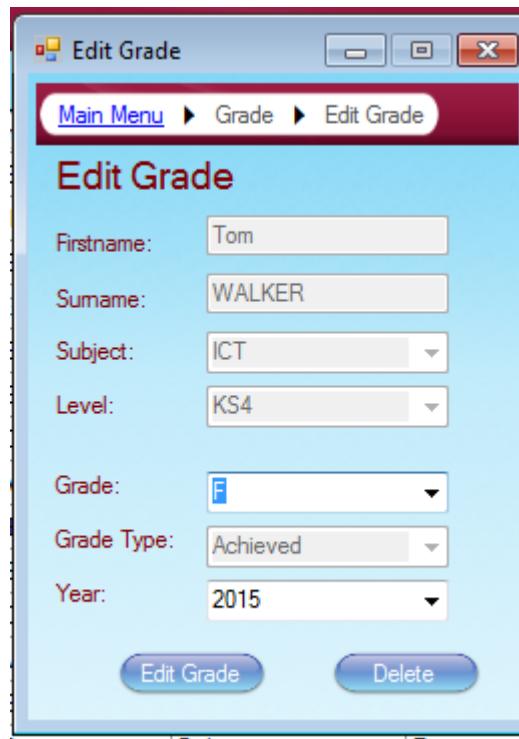
        'the GradeId of the Edit Grade Form is set to the Grade ID that is found from
        'this form
        Edit_Grade.GradeID = GradeID
        'The edit grade form is displayed
        Edit_Grade.Show()
    End If
End Sub
'when the main menu link is clicked then the following forms are either displayed or
closed
Private Sub MainMenu_LinkClicked(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles MainMenu.LinkClicked
    Main_Menu.Show()
    Edit_Grade.Close()
    Add_Student_Grade.Close()
    Me.Close()
End Sub
'this sub is run when the Clear Selection button is clicked
Private Sub Clearselection_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Clearselection.Click
    'clears all controls on this form
    Me.Controls.Clear()
    're-initializes all components of this form
    InitializeComponent()
    'then re-loads this form with all components reset
    Grade_Load(e, e)
End Sub
'when the add grade button is clicked the following forms are either closed or opened
Private Sub Addbutton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Addbutton.Click
    Add_Student_Grade.Show()
    Edit_Grade.Close()
    Me.Close()
End Sub
'deals with the print form button
Private Sub printbutton_Click(sender As System.Object, e As System.EventArgs) Handles
printbutton.Click
    'sets the printer settings to default
    PrintForm.PrinterSettings = PrintDialog.PrinterSettings
    If PrintDialog.ShowDialog() = DialogResult.OK Then
        'allows the user to change print settings
        PrintForm.PrinterSettings = PrintDialog.PrinterSettings
        'changes the orientation of the printed document to landscape
        Me.PrintForm.PrinterSettings.DefaultPageSettings.Landscape = True

```

```
'refreshes the form so that the printdialog is not printed along with the form
Me.Refresh()
'prints the current form
Me.PrintForm.Print()
End If
End Sub
End Class
```

Edit Grade

This form can be used to edit any grad within the database system. A grade ID is passed into this form which determines the selected Grade details that are displayed. The details associated with the grade ID that are passed into this form are displayed within the textbox and combo box fields; these include details such as the name of the student, the subject and level that the grade ID is attributed to. The a user is restricted as to which fields they can alter, they can only change the grade of the record and the year of the record, all other details associated with the record are restricted. When a user is happy with their edited values they can click the edit grade button which will edit the record of the grade ID that is selected and update that record with the new values. The Grade record can also be deleted if the delete button is clicked, in which case a confirmation dialog box will ask the user if they really want to delete the record.



```

arrowlink System.Windows.Forms.Label
arrowlink1 System.Windows.Forms.Label
banner System.Windows.Forms.PictureBox
Deletebtn System.Windows.Forms.Button
Edit System.Windows.Forms.Button
Edit_Grade System.Windows.Forms.Form
Editlabel System.Windows.Forms.LinkLabel
edittitle System.Windows.Forms.Label
Firstname System.Windows.Forms.TextBox
firstnamelabel System.Windows.Forms.Label
Gradefield System.Windows.Forms.ComboBox
Gradelabel System.Windows.Forms.Label
Gradetypefield System.Windows.Forms.ComboBox
Gradetypelabel System.Windows.Forms.Label
LevelField System.Windows.Forms.ComboBox
Levellabel System.Windows.Forms.Label
MainMenu System.Windows.Forms.LinkLabel
Subjectfield System.Windows.Forms.ComboBox
Subjectlabel System.Windows.Forms.Label
SubjectLink System.Windows.Forms.LinkLabel
Surname System.Windows.Forms.TextBox
surnamlabel System.Windows.Forms.Label
Yearfield System.Windows.Forms.ComboBox
Yearlabel System.Windows.Forms.Label

```

Parameter Name	Description
GradeID	This sub uses a get command to fetch the corresponding GradeID of the row the user wants to add a student to and stores it as a private integer, which will be used to reference the correct class row in the database.
Edit_Grade_Load	This code encased within this sub initiates when the form loads. A dataset variable is filled with the corresponding grade values of the GradeID that is passed to the form. The values of the record returned in the dataset are used to set the text properties of the text and combo-box fields on the form.
GradeLink_LinkClicked	When this link label is clicked the user is re-directed to the Grade form. If fields on the form have been edited then the user is asked to confirm their navigation choice.
MainMenu_LinkClicked	When this link label is clicked the user is re-directed to the Main menu form. If fields on the form have been edited then the user is asked to confirm their navigation choice.
editedcheck	This function sets a true or false condition depending on whether the values in any of the entry boxes differ from their original values. If they differ then a false state is returned.
txtvalidate	This function sets a true or false condition depending on whether any textboxes on the form are blank. It also sets the backcolor of textboxes appropriately to indicate to the user that fields have not been completed.
gradevalidate	This function returns a true or false state depending on whether the grade the user has selected cannot be a part of the grade type the user has selected.
IDexists	This function returns a true or false state depending on whether the grade the user wants to delete actually exists within the database system, if a row is returned when searching via the GradeID then the record does exist.
rowexists	This function also returns a true or false state depending on whether there is a record in the grade table of the database matching the user's selection.
Edit_Click	When the edit class button is clicked the record that the user has selected in the database is updated with the values they have entered in the forms entry boxes.
Deletebtn_Click	When the delete button is clicked by the user, and the row exists, then a SQL statement removes the record from the database.

```

'imports the following system code commands so that i can use them appropriately
Imports System.Data.SqlClient
Imports System.Data
Imports System.Configuration
Imports System.Data.DataTable
Public Class Edit_Grade
    'sets the variable that is received from the Grade form as a private variable which
    can be used in any sub within this class
    Private _GradeID As Integer
    'stores a studentID variable within this class
    Dim SID As Integer

    'sets the values of several variables that can be used to check the values of these
    variables later in the code, they can be used in any sub
    Dim Gradecheck As String
    Dim Gradetypecheck As String
    Dim Yearcheck As String
    'creates a property of the variable that is being received
    Public Property GradeID() As Integer
        Get
            'gets the variable from the form that has been specified, in this case the
            grade form
            Return _GradeID
        End Get
        Set(ByVal value As Integer)
            'sets the variable that has been received by the get statement as the
            equivalent variable that will be used in this form
            _GradeID = value
        End Set
    End Property
    'This sub runs when the Edit grade form is initially loaded
    Private Sub Edit_Grade_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Me.Load

        'sets the screen dimensions of the form to automatically generate center screen of
        the users display
        Me.Left = (Screen.PrimaryScreen.WorkingArea.Width - Me.Width) / 2
        Me.Top = (Screen.PrimaryScreen.WorkingArea.Height - Me.Height) / 2
        'calls database connect from the public module
        DatabaseConnect()
        'stores dt as a datatable
        Dim dt As DataTable
        'stores ds as a dataset
        Dim ds As DataSet

        'Uses the Generate combo function of the module to populate a combobox by passing
        the table, field and needempty variables to the Generatecombo function and then returning
        the result in a datatable
        dt = GenerateCombo("GradeType", "GradeTypeID", GradeType", False)
        'sets the datasource of the combobox
        Gradetypefield.DataSource = dt
        'sets the display member of the combobox
        Gradetypefield.DisplayMember = "GradeType"
        'sets the index value of each item in the Gradetype combobox to match their
        corresponding ID's
        Gradetypefield.ValueMember = "GradeTypeID"

        dt = GenerateCombo("DatasetID", "DatasetID.DatasetID,Year", True)
        Yearfield.DataSource = dt
        Yearfield.DisplayMember = "Year"
        Yearfield.ValueMember = "DatasetID.DatasetID"

```

```

dt = GenerateCombo("Subject", "SubjectID, Subject", True)
Subjectfield.DataSource = dt
Subjectfield.DisplayMember = "Subject"
Subjectfield.ValueMember = "SubjectID"

dt = GenerateCombo("Level", "LevelID,Level", True)
LevelField.DataSource = dt
LevelField.DisplayMember = "Level"
LevelField.ValueMember = "LevelID"

dt = GenerateCombo("Grade", "Grade", True)
Gradefield.DataSource = dt
Gradefield.DisplayMember = "Grade"

'the dataset is set as the returned dataset after passing GradeID in to the
Editgradevalues function in the module
ds = Editgradevalues(GradeID)
'the following textbox fields are set to their respective values under their
column headings within the dataset
Firstname.Text = ds.Tables(0).Rows(0).Item("Firstname").ToString()
Surname.Text = ds.Tables(0).Rows(0).Item("Surname").ToString()
Subjectfield.Text = ds.Tables(0).Rows(0).Item("Subject").ToString()
LevelField.Text = ds.Tables(0).Rows(0).Item("Level").ToString()

'the following variables are set to their equivalent values under their respective
headings in the dataset
Gradecheck = ds.Tables(0).Rows(0).Item("Grade").ToString()
Gradetypecheck = ds.Tables(0).Rows(0).Item("GradeType").ToString()
Yearcheck = ds.Tables(0).Rows(0).Item("Year").ToString()
'sets the text of the textboxes as the values of the variables above that have
been set
Gradefield.Text = Gradecheck
Gradetypefield.Text = Gradetypecheck
Yearfield.Text = Yearcheck
End Sub
'this sub runs when the Grade link is clicked
Private Sub GradeLink_LinkClicked(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles SubjectLink.LinkClicked
'the following forms either display or close if edited check is true
If editedcheck() Then
    Grade.Show()
    Add_Student_Grade.Close()
    Me.Close()
    'if edited check is not true then a dialog message displays to the user with a
yes/no option
ElseIf editedcheck() = False Then
    Dim result As Integer = MessageBox.Show("Do you really want to change
window?", "Edited Record", MessageBoxButtons.YesNo, MessageBoxIcon.Warning)
    'if the user clicks yes then the following forms either display or close
    If result = DialogResult.Yes Then
        Grade.Show()
        Add_Student_Grade.Close()
        Me.Close()
        'if the user clicks no then nothing happens
    ElseIf result = DialogResult.No Then
        End If
    End If
End Sub
'this sub runs when the main menu link is clicked and functions in the same manner as
the teacher link above

```

```

Private Sub MainMenu_LinkClicked(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles MainMenu.LinkClicked
    If editedcheck() Then
        Main_Menu.Show()
        Add_Student_Grade.Close()
        Grade.Close()
        Me.Close()
    ElseIf editedcheck() = False Then
        Dim result As Integer = MessageBox.Show("Do you really want to change
window?", "Edited Record", MessageBoxButtons.YesNo, MessageBoxIcon.Warning)
        If result = DialogResult.Yes Then
            Main_Menu.Show()
            Add_Student_Grade.Close()
            Grade.Close()
            Me.Close()
        ElseIf result = DialogResult.No Then
            End If
        End If
    End Sub
    'this function returns a true or false state depending on whether conditions within
the function have been met or not
    Public Function editedcheck() As Boolean
        Call DatabaseConnect()
        'initially the functions state is set to true
        editedcheck = True
        'but if any of the following textbox fields values are not the same as their
        respective stored variables then the functions state is set to false
        If Gradefield.Text <> Gradecheck Then
            editedcheck = False
        End If
        If Gradetypefield.Text <> Gradetypecheck Then
            editedcheck = False
        End If
        If Yearfield.Text <> Yearcheck Then
            editedcheck = False
        End If
    End Function
    'this function returns a true or false state depending on whether conditions within
the function have been met or not
    Public Function txtvalidate() As Boolean
        Call DatabaseConnect()
        'initially the functions state is set to true
        txtvalidate = True

        'if any of the textboxes are left blank then the function returns a false state
        and the background of the textbox field changes to red, otherwise the background of the
        textbox changes to white
        If Gradefield.Text = "" Then
            Gradefield.BackColor = Color.Salmon
            txtvalidate = False
        Else
            Gradefield.BackColor = Color.White
        End If
        If Gradetypefield.Text = "" Then
            Gradetypefield.BackColor = Color.Salmon
            txtvalidate = False
        Else
            Gradetypefield.BackColor = Color.White
        End If
        If Yearfield.Text = "" Then
            Yearfield.BackColor = Color.Salmon
            txtvalidate = False
        End If
    End Function

```

```

    Else
        Yearfield.BackColor = Color.White
    End If

End Function
'this function returns a true or false state depending on whether conditions within
the function have been met or not
Public Function gradevalidate() As Boolean
    Call DatabaseConnect()
    'initially the functions state is set to true
    gradevalidate = True

    'if the levelfield is KS2 and the value of the gradefield is a number( which can
only be applied to KS2 subjects) then this functions state is set to false
    If LevelField.Text = "KS4" And IsNumeric(Gradefield.Text) = True Then
        gradevalidate = False
    End If
    'if the levelfield is KS4 and the value of the gradefield is not a number(which
can only be applied to KS2 subjects) then this functions state is set to false
    If LevelField.Text = "KS2" And IsNumeric(Gradefield.Text) = False Then
        gradevalidate = False
    End If
End Function
'this function returns a true or false state depending on whether the GradeID exists
Public Function IDexists(ByVal GradeID As String) As Boolean
    'stores cmd as a sqlCommand()
    Dim Cmd As New SqlCommand
    'stores da as a SQL dataadapter
    Dim da As New SqlDataAdapter
    'stores dt as a datatable
    Dim dt As New DataTable
    'stores ds as a dataset
    Dim ds As New DataSet
    'connects to the SQL database by calling the DatabaseConnect from the module
    Call DatabaseConnect()

    'sets the SQL statement
    sql = "Select * from Grade where GradeID = " & GradeID & ""
    'sets the connection path as the connectionstring of con from the module
    Cmd.Connection = con
    'sets the command statement as the SQL statement
    Cmd.CommandText = sql
    'the dataadapter then uses this SQL command to Query the database
    da.SelectCommand = Cmd
    'the dataadapter retrieves the query results from the SQL statement after
    connecting to the Database and fills a dataset
    da.Fill(ds, "Grade")
    'the datatable variable is then set to the table result of the dataset variable
    dt = ds.Tables("Grade")
    'if the nubmer of rows in the datatable is one or more then the IDexists state is
    set to true otherwise it is set to false
    If dt.Rows.Count >= 1 Then
        IDexists = True
    Else
        IDexists = False
    End If
End Function
'this function returns a true or false state depending on whether the row exists in
the database, and works in a similar mannar as the function seen above
Public Function rowexists(ByVal SID As Integer) As Boolean
    Dim Cmd As New SqlCommand
    Dim da As New SqlDataAdapter

```

```

Dim dt As New DataTable
Dim ds As New DataSet
Call DatabaseConnect()
sql = "Select DISTINCT * FROM Grade WHERE StudentID = '" & SID & "' AND LevelID =
'" & LevelField.SelectedValue & "' AND SubjectID = '" & Subjectfield.SelectedValue & "' AND Grade =
'" & Gradefield.Text & "' AND GradeTypeID = '" & Gradetypefield.SelectedValue & "' AND DatasetID =
'" & Yearfield.SelectedValue & "';"
Dim cmmnd As New SqlCommand(sql, con)
da.SelectCommand = cmmnd
da.Fill(ds, "temp")
dt = ds.Tables("temp")
If dt.Rows.Count >= 1 Then
    rowexists = True
Else
    rowexists = False
End If
End Function
'this sub runs when the edit button is clicked by the user
Private Sub Edit_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Edit.Click
    Dim da As New SqlDataAdapter
    Dim dt As New DataTable
    Dim ds As New DataSet

    DatabaseConnect()
    'if the textboxes on the form are all not blank then the code continues
    If txtvalidate() = True Then
        'If the GradeID exists then the code continues
        If IDexists(GradeID) = True Then
            'if the grade entered is valid and can be applied to the selected keystage
            then the code continues
            If gradevalidate() = True Then
                'sets the SQL statement
                sql = "UPDATE Grade SET Grade.Grade = @Grade, Grade.GradetypeID =
@GradeT, Grade.DatasetID = @Year WHERE GradeID = " & GradeID & ";"
                'uses cmd as a new SQL command that uses the above SQL statement and
                the connection string of con
                Using cmd = New SqlCommand(sql, con)
                    'sets the parameters of the vlues to be inserted
                    cmd.Parameters.AddWithValue("Grade", Gradefield.Text)
                    cmd.Parameters.AddWithValue("GradeT",
Gradetypefield.SelectedValue)
                    cmd.Parameters.AddWithValue("Year", Yearfield.SelectedValue)
                    'executes the SQL command
                    cmd.ExecuteNonQuery()
                End Using
                'closes connection with the database
                con.Close()
                'informs the user with a message that the record has been updated
                MsgBox("Record Updated", MsgBoxStyle.Information)
            Else
                'if the grade entered is not valid for that Keystage then the
                following error message displays and nothing happens
                MsgBox("That grade cannot be applied to that Subject Level
(Keystage)", MsgBoxStyle.Information, MessageBoxButtons.OK)
            End If
        Else
            ' 'If the GradeID does not exist then the following error message displays
            and nothing happens
            MsgBox("The current GradeID does not exist", MsgBoxStyle.Information,
MessageBoxButtons.OK)
        End If
    End If

```

```

    Else
        'if one or more of the textboxes on the form are blank then the following
        error message displays and nothing happens
        MsgBox("One or more textboxes have not been completed",
        MsgBoxStyle.Information, MessageBoxButtons.OK)
    End If
End Sub
'this sub runs when the delete button is clicked and works in a similar manner to the
code in the above sub
Private Sub Deletebtn_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Deletebtn.Click
    Dim da As New SqlDataAdapter
    Dim dt As New DataTable
    Dim ds As New DataSet
    DatabaseConnect()

    'this block of code sets the StudentID of the record that the user wants to delete
    based on the values of the firstname and surname textboxes
    sql = "Select Distinct StudentID FROM Student WHERE Fristname = '" &
    Fristname.Text & "' AND Surname = '" & Surname.Text & "' "
    Dim cmmnd As New SqlCommand(sql, con)
    da.SelectCommand = cmmnd
    da.Fill(ds, "Student")
    dt = ds.Tables("Student")
    SID = ds.Tables(0).Rows(0).Item("StudentID").ToString()

    If txtvalidate() Then
        If rowexists(SID) = True Then
            'provides a yes/no dialog box asking the user if they really want to
            delete the record
            Dim result As Integer = MessageBox.Show("Do you really want to delete this
            record?", "Delete", MessageBoxButtons.YesNo, MessageBoxIcon.Warning)
            'if the answer is yes then the following code deletes the record from the
            database using the SQL statement below
            If result = DialogResult.Yes Then
                sql = "DELETE FROM Grade WHERE StudentID = '" & SID & "' AND LevelID =
                '" & LevelField.SelectedValue & "' AND SubjectID = '" & Subjectfield.SelectedValue & "'"
                AND Grade = '" & Gradefield.Text & "' AND GradeTypeID = '" & Gradetypefield.SelectedValue
                & "' AND DatasetID = '" & Yearfield.SelectedValue & "';"
                Using cmd = New SqlCommand(sql, con)
                    cmd.ExecuteNonQuery()
                End Using
                con.Close()
                MsgBox("Record Deleted From Database", MsgBoxStyle.Information)
                'the following forms are either displayed or closed
                Grade.Show()
                Me.Close()
                'but if the answer is no nothing happens
            ElseIf result = DialogResult.No Then
                End If
            Else
                'if the rowexists state is not true the appropriate error message is
                displayed
                MsgBox("The Entered Record does not exist", MsgBoxStyle.Information,
                MessageBoxButtons.OK)
            End If
        Else
            'if the txtvalidate state is not true the appropriate error message is
            displayed
            MsgBox("One or more textboxes have not been completed",
            MsgBoxStyle.Information, MessageBoxButtons.OK)

```

```
    End If  
End Sub  
End Class
```

Add Grade

The add grade form allows a user to display all of the grades that a student in the system currently has and then allows the option of adding a new grade to the selected student. A student can be selected by using the filter options which displays a list of students in the data grid matching particular search criteria's. When a student is selected, a list of that student's grades displays in the data grid. Each of these grades is assigned a button, that when clicked loads an edit grade form allowing the record to either be edited or deleted from the database. Furthermore, when a student is selected a new group box displays to the user which allows the addition of a new grade to be assigned to the selected student and inserted into the system's database. A printed copy of this form can also be produced so that a user has a hard copy of all the grades that a student has.

Firstname	Surname	Subject	Grade	Level	GradeType	Year	Grade Details
Hamzah	Finlay	ICT	D	KS4	Achieved	2014	<button>Edit Grade</button>
Hamzah	Finlay	Drama	B	KS4	Achieved	2014	<button>Edit Grade</button>
Hamzah	Finlay	English Language	B	KS4	Achieved	2014	<button>Edit Grade</button>
Hamzah	Finlay	English Literature	B	KS4	Achieved	2014	<button>Edit Grade</button>
Hamzah	Finlay	History	B	KS4	Achieved	2014	<button>Edit Grade</button>
Hamzah	Finlay	French	B	KS4	Achieved	2014	<button>Edit Grade</button>
Hamzah	Finlay	Biology	B	KS4	Achieved	2014	<button>Edit Grade</button>
Hamzah	Finlay	Physics	B	KS4	Achieved	2014	<button>Edit Grade</button>
Hamzah	Finlay	Maths	A*	KS4	Achieved	2014	<button>Edit Grade</button>
Hamzah	Finlay	Business Studies	A	KS4	Achieved	2014	<button>Edit Grade</button>
Hamzah	Finlay	Chemistry	A	KS4	Achieved	2014	<button>Edit Grade</button>
Hamzah	Finlay	English Language	4	KS2	Achieved	2014	<button>Edit Grade</button>
Hamzah	Finlay	Maths	4	KS2	Achieved	2014	<button>Edit Grade</button>

Add_Student_Grade System.Windows.Forms.Form
Addgradebutton System.Windows.Forms.Button
AddGradeGrid System.Windows.Forms.DataGridView
Addgradegroup System.Windows.Forms.GroupBox
Addsearch System.Windows.Forms.Button
banner System.Windows.Forms.PictureBox
Editlabel System.Windows.Forms.LinkLabel
Field System.Windows.Forms.ComboBox
fieldlabel System.Windows.Forms.Label
Firstname System.Windows.Forms.TextBox
firstnamelabel System.Windows.Forms.Label
Formheader System.Windows.Forms.Label
Gradefield System.Windows.Forms.ComboBox
Gradelabel System.Windows.Forms.Label
GradeLink System.Windows.Forms.LinkLabel
Gradetypefield System.Windows.Forms.ComboBox
Gradetypelabel System.Windows.Forms.Label
LevelField System.Windows.Forms.ComboBox
Levellabel System.Windows.Forms.Label
linkarrow1 System.Windows.Forms.Label
linkarrow2 System.Windows.Forms.Label
MainMenu System.Windows.Forms.LinkLabel
printbutton System.Windows.Forms.Button
PrintDialog System.Windows.Forms.PrintDialog
PrintForm Microsoft.VisualBasic.PowerPacks.Printing.PrintForm

SearchDataset System.Windows.Forms.ComboBox
SearchEthnicity System.Windows.Forms.ComboBox
SearchForm System.Windows.Forms.ComboBox
SearchGender System.Windows.Forms.ComboBox
SearchName System.Windows.Forms.TextBox
SearchSNS System.Windows.Forms.ComboBox
SearchStudent System.Windows.Forms.GroupBox
SearchYear System.Windows.Forms.ComboBox
Subjectfield System.Windows.Forms.ComboBox
Subjectlabel System.Windows.Forms.Label
Surname System.Windows.Forms.TextBox
surnamelabel System.Windows.Forms.Label
wherelabel System.Windows.Forms.Label
Yearfield System.Windows.Forms.ComboBox
Yearlabel System.Windows.Forms.Label

Parameter Name	Description
Add_Grade_Load	This sub connects to the database server when the form initially loads and auto-resizes the screen dimensions of the form so that they fit centre screen.
GradeLink_LinkClicked	When this link label is clicked the user is redirected to the Grade form. If fields on the form have been edited then the user is asked to confirm their navigation choice.
editedcheck	This function sets a true or false condition depending on whether the values in any of the entry boxes differ from their original values. If they differ then a false state is returned.
MainMenu_LinkClicked	When this link label is clicked the user is redirected to the Main menu form. If fields on the form have been edited then the user is asked to confirm their navigation choice.
Field_SelectedIndexChanged	Whenever the text value of the filter field is changed this sub determines which objects are either visible or non-visible.
Addsearch_Click	This sub handles the instance the search button is clicked by the user. It queries students that match the user's filter selection and then displays them in the data grid. It also adds a new button column to the data grid.
AddGradeGrid_ContentClick	Handles the instance of a button being clicked on the data grid, if the view grades button is clicked then the data grid displays all the grades that the student on the row the button is clicked has. If the button that is clicked is edit grade then the edit grade form is displayed to the user and the id of the grade the user wants to edit is passed to the edit grade form.
gradevalidate	This function returns a true or false state depending on whether the grade the user has selected cannot be a part of the grade type the user has selected.
Gradeexists	This function also returns a true or false state depending on whether there is a record in the grade table of the database matching the user's selection.
txtvalidate	This function sets a true or false condition depending on whether any textboxes on the form are blank. It also sets the background colour of textboxes appropriately to indicate to the user that fields have not been completed.
Addgradebutton_Click	This sub handles the instance the add grade button is clicked. It is responsible for validating for inserting a new grade into the database

	system that matches the details the user has entered into the entry boxes.
printbutton_Click	This sub allows the user to print a hard copy of the form when the print icon is clicked, additional print dialog options are then displayed allowing the user to choose printer and number of copies amongst other settings.

'Imports the following system code commands so that I can use them appropriately

```

'imports the following system code commands so that I can use them appropriately
Imports System.Data.SqlClient
Imports System.Data
Imports System.Configuration
Imports System.Data.DataTable

Public Class Add_Student_Grade
    'stores a studentID that can be used anywhere within this class
    Dim studentID As Integer
    'stores a validation variable that validates which action to perform later on
    Dim validate6 As Boolean = True
    'this sub runs when the form is initially loaded
    Private Sub Add_Grade_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Me.Load
    'calls database connect from the public module
    DatabaseConnect()
    'sets the screen dimensions of the form to automatically generate center screen of
    'the users display
    Me.Left = (Screen.PrimaryScreen.WorkingArea.Width - Me.Width) / 2
    Me.Top = (Screen.PrimaryScreen.WorkingArea.Height - Me.Height) / 2
    'the datagrid is set to not be visible
    AddGradeGrid.Visible = False
    'clears the current selection of columns in thedatagrid
    AddGradeGrid.Columns.Clear()
End Sub
    'this sub runs when the Grade link is clicked
    Private Sub GradeLink_LinkClicked(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles GradeLink.LinkClicked
        'the following forms either display or close if edited check is true
        If editedcheck() Then
            Grade.Show()
            Edit_Grade.Close()
            Me.Close()
            'if edited check is not true then a dialog message dsiplays to the user with a
            yes/no option
        ElseIf editedcheck() = False Then
            Dim result As Integer = MessageBox.Show("Do you really want to change
            window?", "Edited Record", MessageBoxButtons.YesNo, MessageBoxIcon.Warning)
            'if the user clicks yes then the following forms either display or close
            If result = DialogResult.Yes Then
                Grade.Show()
                Edit_Grade.Close()
                Me.Close()
                'if the user clicks no then nothing happens
            ElseIf result = DialogResult.No Then
                End If
            End If
        End Sub
        'this sub runs when the main menu link is clicked and functions in the same mannar as
        'the teacher link above
        Private Sub MainMenu_LinkClicked(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles MainMenu.LinkClicked
        If editedcheck() Then
            Main_Menu.Show()
            Edit_Grade.Close()
            Grade.Close()
            Me.Close()
        ElseIf editedcheck() = False Then
            Dim result As Integer = MessageBox.Show("Do you really want to change
            window?", "Edited Record", MessageBoxButtons.YesNo, MessageBoxIcon.Warning)
            If result = DialogResult.Yes Then
                Main_Menu.Show()

```

```

        Edit_Grade.Close()
        Grade.Close()
        Me.Close()
    ElseIf result = DialogResult.No Then
        End If
    End If
End Sub
'this function returns a true or false state depending on whether conditions within
the function have been met or not
Public Function editedcheck() As Boolean
    Call DatabaseConnect()
    'initially the functions state is set to true
    editedcheck = True
    'but if any of the following textbox fields values are not the same as their
respective stored variables then the functions state is set to false

    If Subjectfield.Text <> "" Then
        editedcheck = False
    End If
    If LevelField.Text <> "" Then
        editedcheck = False
    End If
    If Gradefield.Text <> "" Then
        editedcheck = False
    End If
    If Yearfield.Text <> "" Then
        editedcheck = False
    End If
End Function
'when the text within the field combobox is changed then the code within this sub is
run
Private Sub Field_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Field.SelectedIndexChanged
    'stores dt as a datatable
    Dim dt As New DataTable
    'sets the wherelabel as visible to the user
    wherelabel.Visible = True

    'This case statement selects which textbox or combobox is visible when the
corrsponding text value of the field combobox is selected and sets all other textboxes or
comboboxes as not visible when a case is selected by the user's mouse click choice
    Select Case Field.Text
        Case ""
            SearchGender.Visible = False
            SearchName.Visible = False
            SearchForm.Visible = False
            SearchYear.Visible = False
            wherelabel.Visible = False
            Addsearch.Visible = False
        Case "Name"
            SearchGender.Visible = False
            SearchName.Visible = True
            SearchForm.Visible = False
            SearchYear.Visible = False
            Addsearch.Visible = True
            wherelabel.Text = "Name:"
        Case "Gender"
            'Uses the Generate combo function of the module to populate a combobox by
passing the table, field and needempty variables to the Generatecombo function and then
returning the result in a datatable
            dt = GenerateCombo("Student", "Gender", False)
            'sets the datasource of the combobox

```

```

        SearchGender.DataSource = dt
        'sets the display member of the combobox

        SearchGender.DisplayMember = "Gender"
        SearchGender.Visible = True
        SearchName.Visible = False
        SearchForm.Visible = False
        SearchYear.Visible = False
        Addsearch.Visible = True
        wherelabel.Text = "Gender:"
    Case "Form"
        'selects the forms based on the user's choice of year
        sql = "SELECT Distinct Form FROM Student WHERE DatasetID = '" &
moduleDataset & "'"
        'stores a SQL command that uses the SQL statement to query the database
        that is found when following the connection string of con
        Using comm As SqlCommand = New SqlCommand(sql, con)
            'stores a SQL dataadareader which can read the contents of the sql
            query from the database
            Dim rs As SqlDataReader = comm.ExecuteReader
            'stores a datatable
            'the datatable is filled with the values that the datareader reads
            dt.Load(rs)

            'sets the displaymemeber of the combobox
            SearchForm.DisplayMember = "Form"
            'sets the datasource of the combobox
            SearchForm.DataSource = dt
        End Using

        SearchGender.Visible = False
        SearchName.Visible = False
        SearchForm.Visible = True
        SearchYear.Visible = False
        Addsearch.Visible = True
        wherelabel.Text = "Form:"
    End Select
End Sub
'this sub runs when the search button is clicked by the user
Private Sub Addsearch_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Addsearch.Click
    'clears the current selection of columns in the studentgrid
    AddGradeGrid.Columns.Clear()
    'the validation variable is set to true
    validate6 = True
    'stores a dataset variable
    Dim ds As New DataSet

    'sets the SQL statement
    sql = "SELECT StudentID, Firstname, Surname, Gender, Form, Year FROM Student"
    'This Case statement is used to alter the SQL query depending on which text is
    selected from the combobox, when a case is selected the above SQL statement and the
    corresponding case 'SQL where statement' are concatenated to form one SQL statement
    Select Case Field.Text
        Case ""
            sql = sql & " ORDER BY StudentID ASC"
        Case "Name"
            sql = sql & " WHERE (Firstname LIKE '%" & SearchName.Text & "%' Or Surname
            LIKE '%" & SearchName.Text & "%') and DatasetID = '" & moduleDataset & "' ORDER BY
            StudentID ASC"
        Case "Gender"
    End Sub

```

```

        sql = sql & " WHERE Gender LIKE '%" & SearchGender.Text & "%'and DatasetID
= '' & moduleDataset & "'ORDER BY StudentID ASC"
        Case "Form"
            sql = sql & " WHERE Form LIKE '%" & SearchForm.Text & "%'and DatasetID =
'" & moduleDataset & "'ORDER BY StudentID ASC"
        End Select

        'the dataset is set after passing the following SQL statement and table into the
GenerateDataset sub of the public module
        ds = generateDataset(sql, "temp")
        'sets the datasrouce of the datagrid
        AddGradeGrid.DataSource = ds
        'grid becomes visible
        AddGradeGrid.Visible = True
        'sets the datamember of the datagrid
        AddGradeGrid.DataMember = "temp"
        AddGradeGrid.Columns("StudentID").Visible = False

        'creates a new variable that stores information for a button column that can be
added to the datagrid
        Dim Columnview As New DataGridViewButtonColumn
        With Columnview
            'sets the header text of the button column
            .HeaderText = "Grade Details"
            'sets the name of the button column
            .Name = "Details"
            'sets the text displayed on the buttons of the button column in the datagrid
            .Text = "View Grades"
            .UseColumnTextForButtonValue = True
        End With

        'adds the button column to the datagrid
        AddGradeGrid.Columns.Add(Columnview)
        'auto resizes the column widths of the datagrid
        AddGradeGrid.AutoResizeColumns()
    End Sub

    'this sub runs when a button that is part of the datagrid is clicked by the user
    Private Sub AddGradeGrid_ContentClick(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.DataGridViewCellEventArgs) Handles AddGradeGrid.CellClick
        'two differnet datasets are stored
        Dim ds As New DataSet
        Dim ds1 As New DataSet
        Dim dt As DataTable
        'stores a GRADEID
        Dim GradeID As String

        'if the index of the button clicked is 6 and the validate property is true then
        'the code within this if statement runs
        If e.ColumnIndex = 6 And validate6 = True Then
            'the studentID variable is set as the STudentID that is on the same row as the
button clicked by the user
            studentID = AddGradeGrid.Rows(e.RowIndex).Cells("StudentID").Value
            'clears the current selection of columns in the datagrid
            AddGradeGrid.Columns.Clear()

            'works like the example above in the addsearch sub and displays the results of
            'the SQL statement query below in addgradegrid which is then set as visible
            sql = "SELECT DISTINCT Grade.GradeID, Student.Firstname, Student.Surname,
Subject.Subject, Grade.Grade, [Level].[Level], GradeType.GradeType, DatasetID.Year FROM
StudentGroup INNER JOIN Class ON StudentGroup.ClassID = Class.ClassID INNER JOIN Grade
INNER JOIN[Level] ON Grade.LevelID = [Level].LevelID INNER JOIN GradeType ON
Grade.GradeTypeID = GradeType.GradeTypeID INNER JOIN Student ON Grade.StudentID =

```

```

Student.StudentID ON StudentGroup.StudentID = Student.StudentID INNER JOIN Subject ON
Grade.SubjectID = Subject.SubjectID INNER JOIN DatasetID ON Grade.DatasetID =
DatasetID.DatasetID WHERE Student.StudentId = '' & studentID & '' ORDER BY GradeID ASC"
    ds = generateDataset(sql, "temp")
    AddGradeGrid.DataSource = ds
    AddGradeGrid.Visible = True
    AddGradeGrid.DataMember = "temp"
    AddGradeGrid.Columns("GradeID").Visible = False
    'auto resizes the column selection
    AddGradeGrid.AutoResizeColumns()

    'add grade group and all of its controls become visible to the user
    Addgradegroup.Visible = True

    'Uses the Generate combo function of the module to populate a combobox by
    passing the table, field and needempty variables to the Generatecombo function and then
    returning the result in a datatable
    dt = GenerateCombo("GradeType", "GradeTypeID", GradeType, False)
    'sets the datasource of the combobox
    Gradetypefield.DataSource = dt
    'sets the display member of the combobox
    Gradetypefield.DisplayMember = "GradeType"
    'the index of each item in the combobox is set as their corresponding ID from
the database
    Gradetypefield.ValueMember = "GradeTypeID"

    dt = GenerateCombo("DatasetID", "DatasetID.DatasetID,Year", True)
    Yearfield.DataSource = dt
    Yearfield.DisplayMember = "Year"
    Yearfield.ValueMember = "DatasetID.DatasetID"

    dt = GenerateCombo("Subject", "SubjectID, Subject", True)
    Subjectfield.DataSource = dt
    Subjectfield.DisplayMember = "Subject"
    Subjectfield.ValueMember = "SubjectID"

    dt = GenerateCombo("Level", "LevelID,Level", True)
    LevelField.DataSource = dt
    LevelField.DisplayMember = "Level"
    LevelField.ValueMember = "LevelID"

    dt = GenerateCombo("Grade", "Grade", True)
    Gradefield.DataSource = dt
    Gradefield.DisplayMember = "Grade"

    'creates a new variable that stores information for a button column that can
be added to the datagrid
    Dim Columnnedit As New DataGridViewButtonColumn
    With Columnnedit
        'sets the header text of the button column
        .HeaderText = "Grade Details"
        'sets the name of the button column
        .Name = "Details"
        'sets the text displayed on the buttons of the button column in the
datagrid
        .Text = "Edit Grade"
        .UseColumnTextForButtonValue = True
    End With
    'if the number of columns in the datagrid is 8 then the button column is added
to the datagrid
    If AddGradeGrid.Columns.Count = 8 Then
        AddGradeGrid.Columns.Add(Columnnedit)

```

```

    End If

    'resizes all columns in the datagrid
    AddGradeGrid.AutoResizeColumns()

    'the dataset1 is set as the returned dataset after passing studentid into the
    addgradevalues of the module
    ds1 = Addgradevalues(studentID)
    'the firstname and surname textboxes text are set as their corresponding
    values in the dataset underneath their respective headings
    Firstname.Text = ds.Tables(0).Rows(0).Item("Firstname").ToString()
    Surname.Text = ds.Tables(0).Rows(0).Item("Surname").ToString()
    'the validate state is set to false
    validate6 = False
End If

'if the button clicked is on the eighth column then this if statement runs
If e.ColumnIndex = 8 Then
    Edit_Grade.Close()
    'GradeID is set to its corresponding value in thedatagrid underneath its
    respective heading, and on the same row as the button clicked
    GradeID = AddGradeGrid.Rows(e.RowIndex).Cells("GradeID").Value

    'passes the gradeID of this form to the EditGrade form
    Edit_Grade.GradeID = GradeID
    Edit_Grade.Show()
End If

End Sub
' 'this function returns a true or false state depending on whether conditions within
the function have been met or not
Public Function gradevalidate() As Boolean
    Call DatabaseConnect()
    'initially the functions state is set to true
    gradevalidate = True
    'if the levelfield is KS2 and the value of the gradefield is a number( which can
    only be applied to KS2 subjects) then this functions state is set to false
    If LevelField.Text = "KS4" And IsNumeric(Gradefield.Text) = True Then
        gradevalidate = False
    End If
    'if the levelfield is KS4 and the value of the gradefield is not a number(which
    can only be applied to KS2 subjects) then this functions state is set to false
    If LevelField.Text = "KS2" And IsNumeric(Gradefield.Text) = False Then
        gradevalidate = False
    End If
End Function
'this function returns a true or false state depending on whether the row exists in
the database
Public Function Gradeexists() As Boolean
    Call DatabaseConnect()
    'the functions state is initially set to false
    Gradeexists = False
    'stores cmd as a sqlCommand()
    Dim Cmd As New SqlCommand
    'stores da as a SQL dataadapter
    Dim da As New SqlDataAdapter
    'stores dt as a datatable
    Dim dt As New DataTable
    'stores ds as a dataset
    Dim ds As New DataSet
    'connects to the SQL database by calling the DatabaseConnect from the module
    Call DatabaseConnect()

```

```

'sets the SQL statement
sql = "SELECT DISTINCT Grade.GradeID, Student.Firstname, Student.Surname,
Subject.Subject, Grade.Grade, [Level].[Level], GradeType.GradeType, DatasetID.Year FROM
StudentGroup INNER JOIN Class ON StudentGroup.ClassID = Class.ClassID INNER JOIN Grade
INNER JOIN[Level] ON Grade.LevelID = [Level].LevelID INNER JOIN GradeType ON
Grade.GradeTypeID = GradeType.GradeTypeID INNER JOIN Student ON Grade.StudentID =
Student.StudentID ON StudentGroup.StudentID = Student.StudentID INNER JOIN Subject ON
Grade.SubjectID = Subject.SubjectID INNER JOIN DatasetID ON Grade.DatasetID =
DatasetID.DatasetID WHERE Student.StudentId = '" & studentID & "' AND Grade.SubjectID = ''"
& Subjectfield.SelectedValue & "' AND Grade.LevelID = '" & LevelField.SelectedValue & "'"
AND Grade.GradeTypeID = '" & GradeTypefield.SelectedValue & "' ORDER BY GradeID ASC"
'stores a SQL command that uses the SQL statement to query the database that is
found when following the connection string of con
Dim cmmnd As New SqlCommand(sql, con)
'the dataadapter then uses this SQL command to Query the database
da.SelectCommand = cmmnd
'the dataadapter retrieves the query results from the SQL statement after
connecting to the Database and fills a dataset
da.Fill(ds, "temp")
'the datatable variable is then set to the table result of the dataset variable
dt = ds.Tables("temp")

'if the number of rows in the datatable is one or more then the IDExists state is
set to false otherwise it is set to true
If dt.Rows.Count >= 1 Then
    Gradeexists = True
Else
    Gradeexists = False
End If
End Function
'this function returns a true or false state depending on whether conditions within
the function have been met or not
Public Function txtvalidate() As Boolean
    Call DatabaseConnect()
    'initially the functions state is set to true
    txtvalidate = True
    'if any of the textboxes are left blank then the function returns a false state
    'and the background of the textbox field changes to red, otherwise the background of the
    'textbox changes to white
    If Subjectfield.Text = "" Then
        Subjectfield.BackColor = Color.Salmon
        txtvalidate = False
    Else
        Subjectfield.BackColor = Color.White
    End If
    If LevelField.Text = "" Then
        LevelField.BackColor = Color.Salmon
        txtvalidate = False
    Else
        LevelField.BackColor = Color.White
    End If
    If GradeTypefield.Text = "" Then
        GradeTypefield.BackColor = Color.Salmon
        txtvalidate = False
    Else
        GradeTypefield.BackColor = Color.White
    End If
    If Gradefield.Text = "" Then
        Gradefield.BackColor = Color.Salmon
        txtvalidate = False
    Else
        Gradefield.BackColor = Color.White
    End If

```

```

        Gradetypefield.BackColor = Color.White
    End If
    If Yearfield.Text = "" Then
        Yearfield.BackColor = Color.Salmon
        txtvalidate = False
    Else
        Yearfield.BackColor = Color.White
    End If
End Function
'this sub runs when the add button is clicked by the user
Private Sub Addgradebutton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Addgradebutton.Click
    Dim da As New SqlDataAdapter
    Dim dt As New DataTable
    Dim ds As New DataSet
    'stores the GradeID of the record that will be inserted
    Dim gID As Integer
    'sets the SQL satatement
    sql = "Select TOP 1 GradeID FROM Grade ORDER BY GradeID DESC"
    'stores cmmnd as a SQL command that uses the above SQL statement and the connection
    string of con
    Dim cmmnd As New SqlCommand(sql, con)
    'sets the dataadapter select command as cmmnd
    da.SelectCommand = cmmnd
    'the dataadapter retrieves the query results from the SQL statement after
    connecting to the Database and fills a dataset
    da.Fill(ds, "Grade")
    'the datatable variable is then set to the table result of the dataset variable
    dt = ds.Tables("Grade")
    'The GradeID is then set as the value of the highest GradeID found from the Query
    and then incremented by 1, the SQL statement takes into account removing grades so that
    inserting a record will use the first available GradeID
    gID = ds.Tables(0).Rows(0).Item("GradeID").ToString() + 1
    DatabaseConnect()

    'if the textboxes have all been filled in then the code continues
    If txtvalidate() = True Then
        'if the selected grade can be applied to the selectedkeystage then the code
continues
        If gradevalidate() = True Then
            'if a grade does not already exist for the selected student the code
continues
            If Gradeexists() = False Then
                'sets the SQL statement
                sql = "INSERT INTO Grade (GradeID, StudentID, SubjectID, LevelID,
Grade, GradeTypeID, DatasetID) VALUES (@GradeID, @StudentID, @SubjectID, @LevelID, @Grade,
@GradeT, @Year)"
                'uses cmd as a new SQL command that uses the above SQL statement and
                the connection string of con
                Using cmd = New SqlCommand(sql, con)
                    'sets the parameters of the vlues to be inserted
                    cmd.Parameters.AddWithValue("GradeID", gID)
                    cmd.Parameters.AddWithValue("StudentID", studentID)
                    cmd.Parameters.AddWithValue("SubjectID",
Subjectfield.SelectedValue)
                    cmd.Parameters.AddWithValue("LevelID", LevelField.SelectedValue)
                    cmd.Parameters.AddWithValue("Grade", Gradefield.Text)
                    cmd.Parameters.AddWithValue("GradeT",
Gradetypefield.SelectedValue)
                    cmd.Parameters.AddWithValue("Year", Yearfield.SelectedValue)
                    cmd.ExecuteNonQuery()
                End Using
            End If
        End If
    End If
End Sub

```

```

    'closes the database connection
    con.Close()
    'notifies the user with the appropriate message
    MsgBox("Record Added", MsgBoxStyle.Information)

        'the following code is similar to the code in the addgradeonclick sub
and essentially, refreshed the datagrid to show the new record
        AddGradeGrid.Columns.Clear()
        sql = "SELECT DISTINCT Grade.GradeID, Student.Firstname,
Student.Surname, Subject.Subject, Grade.Grade, [Level].[Level], GradeType.GradeType,
DatasetID.Year FROM StudentGroup INNER JOIN Class ON StudentGroup.ClassID = Class.ClassID
INNER JOIN Grade INNER JOIN[Level] ON Grade.LevelID = [Level].LevelID INNER JOIN GradeType
ON Grade.GradeTypeID = GradeType.GradeTypeID INNER JOIN Student ON Grade.StudentID =
Student.StudentID ON StudentGroup.StudentID = Student.StudentID INNER JOIN Subject ON
Grade.SubjectID = Subject.SubjectID INNER JOIN DatasetID ON Grade.DatasetID =
DatasetID.DatasetID WHERE Student.StudentId = '" & studentID & "'"
        ds = generateDataset(sql, "temp")
        AddGradeGrid.DataSource = ds
        AddGradeGrid.Visible = True
        AddGradeGrid.DataMember = "temp"
        AddGradeGrid.AutoResizeColumns()
        Dim Columnnedit As New DataGridViewButtonColumn
        With Columnnedit
            .HeaderText = "Grade Details"
            .Name = "Details"
            .Text = "Edit Grade"
            .UseColumnTextForButtonValue = True
        End With
        If AddGradeGrid.Columns.Count = 8 Then
            AddGradeGrid.Columns.Add(Columnnedit)
        End If
        AddGradeGrid.AutoResizeColumns()
    Else
        'if a grade for the selected student already exists for the selected
subject the following error message displays and nothing happens
        MsgBox("A grade is already Assigned to that subject",
MsgBoxStyle.Information, MessageBoxButtons.OK)
    End If
    Else
        'if the selected grade can not be applied to the selectedkeystage the
following error message displays and nothing happens
        MsgBox("That grade cannot be applied to that Subject Level (Keystage)",
MsgBoxStyle.Information, MessageBoxButtons.OK)
    End If
    Else
        'if one or more textboxes have not been completed the following error message
displays and nothing happens
        MsgBox("One or more textboxes have not been completed",
MsgBoxStyle.Information, MessageBoxButtons.OK)
    End If
End Sub
'deals with the print form button
Private Sub printbutton_Click(sender As System.Object, e As System.EventArgs) Handles
printbutton.Click
    'sets the printer settings to default
    PrintForm.PrinterSettings = PrintDialog.PrinterSettings
    If PrintDialog.ShowDialog() = DialogResult.OK Then
        'allows the user to change print settings
        PrintForm.PrinterSettings = PrintDialog.PrinterSettings
        'changes the orientation of the printed document to landscape
        Me.PrintForm.PrinterSettings.DefaultPageSettings.Landscape = True
        'refreshes the form so that the printdialog is not printed along with the form

```

```
    Me.Refresh()
    'prints the current form
    Me.PrintForm.Print()
End If
End Sub
End Class
```

Grade Predictions

From this form a user can view and display predicted grades in a series of different ways. The predicted grades are all generated dynamically, so that if an existing grade is altered or new grades are entered into the system the predicted grades will alter appropriately in response. All of an individuals predicted grades can be viewed, a whole form's predicted grades can be viewed, a class's predicted grades can be viewed or an entire year groups predicted grades can be viewed. Once a grid of predicted grades has been displayed a new button appears which allows the user to create a spread sheet file that can be manipulated within spread sheet programs such as Microsoft Excel. This is done by converting the contents of a data grid into a CSV. A printed copy of this form can be produced for further use and further reference.

The screenshot shows a Windows application window titled "Grade Predictions". The main area contains a data grid titled "Predicted Grades" with columns for Firstname, Surname, Art, Biology, Business Studies, Chemistry, Computing, Drama, Economics, English Language, and English Literature. The grid lists numerous student names with their predicted grades in each subject. On the left side of the form, there are several dropdown menus and buttons for filtering and searching. At the bottom, there are buttons for "ClearSelection" and "Create Spreadsheet file".

Firstname	Surname	Art	Biology	Business Studies	Chemistry	Computing	Drama	Economics	English Language	English Literature
William	ABERCROMBIE	B	C	B	C	C	B	B	B	B
William	AKRAM	B	D	B	D	C	C	B	C	C
William	ALLISON	C	D	C	D	D	C	C	C	C
Will	ALTOFT	B	D	C	D	C	C	C	C	C
Tristan	ASHWORTH	C	D	C	D	D	D	C	D	D
Toby	BAIN	E	F	E	F	F	E	E	F	F
Timothy	BALL-WOOD	A	C	A	C	B	A	A	B	B
Thomas	BARBOUR	B	C	B	C	C	B	B	B	B
Thomas	BARKE	A	C	B	C	C	B	B	B	B
Summer	BELL	A	C	A	C	B	B	A	B	B
Sophie	BIRKS	C	D	C	D	D	C	C	D	D
Sophie	BOND	B	C	B	C	C	B	B	B	B
Simon	BRAY	A	C	A	C	B	B	A	B	B
Samuel	BRENTNALL	B	C	B	C	C	B	B	B	B
Sam	BRODERICK	B	C	B	C	C	B	B	B	B
Rosie	BROWN	A	B	A	B	B	A	A	A	A
Robert	BYWATER	C	D	C	D	D	D	C	D	D
Rachel	CANNON	A	C	A	C	B	B	A	B	B
Polly	CANNON	A	C	B	C	C	B	B	B	B
Philip	CARTER	A*	B	A*	B	A	A	A*	A	A
Philip	CASTLEDINE	B	D	C	D	C	C	C	C	C
Paige	CHAMBERS	B	C	B	C	C	B	B	B	B
Olivia	CHAMBERS	A	C	A	C	B	A	A	B	B
Olivia	CLARK	A	C	A	C	B	B	A	B	B
Olivia	CLAYTON	A*	B	A*	B	A	A	A*	A	A
Tom	CLOUGH	B	C	B	C	C	B	B	B	B
Nikisha	CLOUGH	A	C	A	C	B	B	A	B	B

banner System.Windows.Forms.PictureBox
Classeslink System.Windows.Forms.LinkLabel
Classgroupfield System.Windows.Forms.ComboBox
Classlabel System.Windows.Forms.Label
ClassPredictiongroup System.Windows.Forms.GroupBox
Classesearch System.Windows.Forms.Button
classyearfield System.Windows.Forms.ComboBox
classyearlabel System.Windows.Forms.Label
Clearselection System.Windows.Forms.Button
createcsv System.Windows.Forms.Button
Field System.Windows.Forms.ComboBox
fieldlabel System.Windows.Forms.Label
formfield System.Windows.Forms.ComboBox
formgroup System.Windows.Forms.GroupBox
formlabel System.Windows.Forms.Label
grouppredictionsgrid System.Windows.Forms.DataGridView
labelform System.Windows.Forms.Label
labelyear System.Windows.Forms.Label
linkarrow1 System.Windows.Forms.Label
MainMenu System.Windows.Forms.LinkLabel
Predictedgradesgrid System.Windows.Forms.DataGridView
Predictions System.Windows.Forms.Form

printbutton System.Windows.Forms.Button
PrintDialog System.Windows.Forms.PrintDialog
PrintForm Microsoft.VisualBasic.PowerPacks.Printing.PrintForm
SaveFileDialog System.Windows.Forms.SaveFileDialog
SearchDataset System.Windows.Forms.ComboBox
SearchEthnicity System.Windows.Forms.ComboBox
SearchForm System.Windows.Forms.ComboBox
SearchGender System.Windows.Forms.ComboBox
Searchgroup System.Windows.Forms.Button
SearchName System.Windows.Forms.TextBox
SearchSNS System.Windows.Forms.ComboBox
SearchYear System.Windows.Forms.ComboBox
selectstudent System.Windows.Forms.Button
studentgroup System.Windows.Forms.GroupBox
SubjectField System.Windows.Forms.ComboBox
subjectlabel System.Windows.Forms.Label
Viewpredictionsgrid System.Windows.Forms.DataGridView
wherelabel System.Windows.Forms.Label
yearcombo System.Windows.Forms.ComboBox
yeargroup System.Windows.Forms.GroupBox
yearlabel System.Windows.Forms.Label
yearsearch System.Windows.Forms.Button

Parameter Name	Description
Predicted_Grades_Load	This sub connects to the database server when the form initially loads and dynamically generates the list properties of all the combo boxes on the form and sets their value members.
formfieldgenerate	This sub generates the list values of the form combo box field dynamically.
Clearselection_Click	Creates a new instance of every object on the form (essentially re-loads the form).
MainMenu_LinkClicked	When the main menu link label is clicked all open forms close and the main menu displays.
Field_SelectedIndexChanged	Whenever the text value of the filter field is changed this sub determines which objects are either visible or non-visible.
Search_Click	This sub handles the instance the search button that is part of viewing an individual student's predictions is clicked. It queries the database using the user filter selection and displays the results of the query in the data grid.
Predictedgradesgrid_CellContentClick	Handles the click event of the predicted grades grid for an individual student's grades. If the view grades button is clicked then the database is queried using the Student ID of the selected student and their grades are displayed in the data grid, an average point score of that student's grades is then calculated. If the Predicted Grades button is clicked then a predicted grade for every A2 subject is generated and displayed in a separate data grid.
Yearfield_SelectedIndexChanged	This sub sets a series of objects to either be visible of or false depending on the state of the year field.
Searchgroup_Click	This sub generates a SQL statement that searches a Form group based on the user's selection. This SQL statement is then passed into the group predictions sub.
yearsearch_Click	This sub generates a SQL statement that searches a Year group based on the user's selection. This SQL statement is then passed into the group predictions sub.
Classsearch_Click	This sub generates a SQL statement that searches a Class group based on the user's selection. This SQL statement is then passed into the group predictions sub.
grouppredictions	This sub is responsible for generating predicted grades for every A2 subject for every student and for their individual average point scores based on their previously achieved grades. These predicted

	grades are then written to a data table which is then displayed as the data source of the data grid.
btnExport_Click	Handles the instance the create spread sheet file button is clicked. Writes a CSV file string of all the contents in the selected data grid and then allows the user to save the CSV file string as a CSV file using a save dialog box.
printbutton_Click	This sub allows the user to print a hard copy of the form when the print icon is clicked, additional print dialog options are then displayed allowing the user to choose printer and number of copies amongst other settings.
yearcombo_SelectedIndexChanged	Handles the instance that the year combo field is altered and accordingly sets other objects as either visible or false to the user.
SubjectField_SelectedIndexChanged	Handles the instance that the subject field is altered and accordingly sets other objects as either visible or false to the user. It also validates whether fields have been completed and if so generates the Class field.
Classfieldgenerate	This sub generates the list values of the class combo box field dynamically.
classyearfield_SelectedIndexChanged	Handles the instance that the class year combo field is altered and accordingly sets other objects as either visible or false to the user. It also validates whether fields have been completed and if so generates the Class field.
Classgroupfield_SelectedIndexChanged	Handles the instance that the class group field is altered and accordingly sets other objects as either visible or false to the user.

```

Imports the following system code commands so that I can use them appropriately
Imports System.Data.SqlClient
Imports System.Data
Imports System
Imports System.IO
Imports System.Text
Imports System.Configuration
Imports System.Data.DataTable
Public Class Predictions
    'stores a studentID and an averagepointscore variable that can only be used within
    this class
    Dim StudentID As Integer
    Dim averagepointscore As Double
    'this sub runs when the main menu is clicked

    Private Sub MainMenu_LinkClicked(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles MainMenu.LinkClicked
        'displays the main menu and closes this form
        Main_Menu.Show()
        Me.Close()
    End Sub
    'this sub runs when the form initially loads
    Private Sub Predicted_Grades_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Me.Load
        'runs the databaseconnect sub from the public module
        DatabaseConnect()
        'datagrid is not visible
        Predictedgradesgrid.Visible = False
        'the current selection of columns in datagrid are cleared
        Predictedgradesgrid.Columns.Clear()

        Dim dt As DataTable

        'Uses the Generate combo function of the module to populate a combobox by passing
        the table, field and needempty variables to the Generatecombo function and then returning
        the result in a datatable
        dt = GenerateCombo("Subject", "SubjectID, Subject", True)
        'sets the datasource of the combobox
        SubjectField.DataSource = dt
        'sets the display member of the combobox
        SubjectField.DisplayMember = "Subject"
        'the index of each item in the combobox is set as their corresponding ID from the
        database
        SubjectField.ValueMember = "SubjectID"

        dt = GenerateCombo("DatasetID", "DatasetID.DatasetID,Year", True)
        classyearfield.DataSource = dt
        classyearfield.DisplayMember = "Year"
        classyearfield.ValueMember = "DatasetID.DatasetID"

        dt = GenerateCombo("DatasetID", "DatasetID.DatasetID,Year", True)
        Yearfield.DataSource = dt
        Yearfield.DisplayMember = "Year"
        Yearfield.ValueMember = "DatasetID.DatasetID"

        dt = GenerateCombo("DatasetID", "DatasetID.DatasetID,Year", True)
        yearcombo.DataSource = dt
        yearcombo.DisplayMember = "Year"
        yearcombo.ValueMember = "DatasetID.DatasetID"
    End Sub
    'when the text within the field combobox is changed then the code within this sub is
run

```

```
Private Sub Field_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Field.SelectedIndexChanged
    'stores dt as a datatable
    Dim dt As DataTable

    'sets the wherelabel as visible to the user
    wherelabel.Visible = True
    'This case statement selects which textbox or combobox is visible when the
    corresponding text value of the field combobox is selected and sets all other textboxes or
    comboboxes as not visible when a case is selected by the user's mouse click choice
    Select Case Field.Text
        Case ""
            SearchGender.Visible = False
            SearchName.Visible = False
            SearchEthnicity.Visible = False
            SearchSNS.Visible = False
            SearchForm.Visible = False
            SearchYear.Visible = False
            SearchDataset.Visible = False
            wherelabel.Visible = False
            selectstudent.Visible = False
        Case "Name"
            SearchGender.Visible = False
            SearchName.Visible = True
            SearchEthnicity.Visible = False
            SearchSNS.Visible = False
            SearchForm.Visible = False
            SearchYear.Visible = False
            SearchDataset.Visible = False
            selectstudent.Visible = True
            wherelabel.Text = "Name:"
        Case "Gender"
            dt = GenerateCombo("Student", "Gender", False)
            SearchGender.DataSource = dt
            SearchGender.DisplayMember = "Gender"

            SearchGender.Visible = True
            SearchName.Visible = False
            SearchEthnicity.Visible = False
            SearchSNS.Visible = False
            SearchForm.Visible = False
            SearchYear.Visible = False
            SearchDataset.Visible = False
            selectstudent.Visible = True
            wherelabel.Text = "Gender:"
        Case "Form"
            dt = GenerateCombo("Student", "Form", False)
            SearchForm.DataSource = dt
            SearchForm.DisplayMember = "Form"

            SearchGender.Visible = False
            SearchName.Visible = False
            SearchEthnicity.Visible = False
            SearchSNS.Visible = False
            SearchForm.Visible = True
            SearchYear.Visible = False
            SearchDataset.Visible = False
            selectstudent.Visible = True
            wherelabel.Text = "Form:"
        Case "Year"
            dt = GenerateCombo("Student", "Year", False)
            SearchYear.DataSource = dt
```

```

        SearchYear.DisplayMember = "Year"

        SearchGender.Visible = False
        SearchName.Visible = False
        SearchEthnicity.Visible = False
        SearchSNS.Visible = False
        SearchForm.Visible = False
        SearchYear.Visible = True
        SearchDataset.Visible = False
        selectstudent.Visible = True
        wherelabel.Text = "Year:"

    End Select
End Sub
'When the search button is clicked the code within the following sub is run
Private Sub Search_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles selectstudent.Click
Viewpredictionsgrid.Visible = False
'clears all columns from the predictedgradesgrid
Predictedgradesgrid.Columns.Clear()
'stores a dataset variable
Dim ds As New DataSet

grouppredictionsgrid.Visible = False
createcsv.Visible = False

'sets the SQL statement
sql = "SELECT StudentID, Firstname, Surname, Gender, Form, Year FROM Student"
'This Case statement is used to alter the SQL query depending on which text is
selected from the combobox, when a case is selected the above SQL statement and the
corresponding case 'SQL where statement' are concatenated to form one SQL statement
Select Case Field.Text
    Case ""
        sql = sql & " ORDER BY StudentID ASC"
    Case "Name"
        sql = sql & " WHERE Firstname LIKE '%" & SearchName.Text & "%' Or Surname
LIKE '%' & SearchName.Text & "%' ORDER BY StudentID ASC"
    Case "Gender"
        sql = sql & " WHERE Gender LIKE '%" & SearchGender.Text & "%' ORDER BY
StudentID ASC"
    Case "Form"
        sql = sql & " WHERE Form LIKE '%" & SearchForm.Text & "%' ORDER BY
StudentID ASC"
    Case "Year"
        sql = sql & " WHERE Year LIKE '%" & SearchYear.Text & "%' ORDER BY
StudentID ASC"
End Select

'the dataset is set after passing the following SQL statement and table into the
GenerateDataset sub of the public module
ds = generateDataset(sql, "temp")
'sets the datasource of the datagrid
Predictedgradesgrid.DataSource = ds
'sets the datagrid as visible
Predictedgradesgrid.Visible = True
'sets the datamember of the datagrid
Predictedgradesgrid.DataMember = "temp"
'hides the StudentID column from the user
Predictedgradesgrid.Columns("StudentID").Visible = False
'creates a new variable that stores information for a button column that can be
added to the datagrid
Dim Columnview As New DataGridViewButtonColumn
With Columnview

```

```

'sets the header text of the button column
.HeaderText = "Student'S GCSE Grades"
'sets the name of the button column
.Name = "Details"
'sets the text displayed on the buttons of the button column in the datagrid
.Text = "View Grades"
.UseColumnTextForButtonValue = True
End With

'adds the button column to thedatagrid
Predictedgradesgrid.Columns.Add(Columnview)
'auto resizes the column widths of thedatagrid
Predictedgradesgrid.AutoResizeColumns()
End Sub
'this sub runs when a button cell that is part of thedatagrid is clicked
Private Sub Predictedgradesgrid_CellContentClick(ByVal sender As System.Object, ByVal e As System.Windows.Forms.DataGridViewCellEventArgs) Handles Predictedgradesgrid.CellContentClick
    'stores two datasets
    Dim ds As New DataSet
    Dim ds1 As New DataSet
    'stores a datatable
    Dim dt As New DataTable
    'stores an index that will be used in my for next statement
    Dim i As Integer
    'stores a predictedvalue as a double precision floating point number
    Dim predictedvalue As Double
    'stores the total of all the predicted values as a double precision floating point
number
    Dim currentpredictedtotal As Double = 0

    grouppredictionsgrid.Visible = False
    createcsv.Visible = False

    'if the cell button clicked by the user is part of the 8th column index then the
the code within this if statement runs
    If e.ColumnIndex = 8 Then
        'stores a datatable
        Dim dt1 As New DataTable
        'stores the actual predicted grade
        Dim predictedgrade As String
        'stores the subject text
        Dim subject As String

        'adds the to columns to the datatable
        dt.Columns.Add("Subject")
        dt.Columns.Add("Predicted A2 Grade")

        'queries all of the subjects and places them in alphabetical order
        sql = "SELECT DISTINCT Subject FROM Subject Order By Subject ASC"
        'the dataset is set after passing the following SQL statement and table into
the GenerateDataset sub of the public module
        ds = generateDataset(sql, "temp")
        'the dt1 datatable is filled with the contents of the dataset
        dt1 = ds.Tables(0)

        'this for next statement runs for every row in the dt1 datatable
        For i = 0 To dt1.Rows.Count - 1
            'the subject variable (text) is set as text under the subject heading in
the datatable that is on the same row as the value of the index i
            subject = dt1.Rows(i)("Subject").ToString()

```

```

        'the predicted grade is received after passing the subject text value, the
index position i, the datatable and the averagepointscore into the A2predictedgrade
function in the public module
        predictedgrade = A2predictedgrade(subject, i, dt1, averagepointscore)
        'adds the subject and predicted grade into the dt datatable into he next
available row
        dt.Rows.Add(subject, predictedgrade)
        'repeats for each row
    Next
    'sets the datasource of the viewpredictionsgrid as dt and sets it as visible
to the user
    Viewpredictionsgrid.DataSource = dt
    Viewpredictionsgrid.Visible = True

End If

'if the cell button clicked by the user is part of the 6th column index then the
the code within this if statement runs
If e.ColumnIndex = 6 Then
    'stores GCSEweighting as a double precision floating point number
    Dim GCSEgradeweighting As Double
    'the StudentID variable is set as the StudentID value underneath the StudentID
column header and is on the same row as the button clicked by the user
    StudentID = Predictedgradesgrid.Rows(e.RowIndex).Cells("StudentID").Value
    Predictedgradesgrid.Columns.Clear()

    'this variable is used to count the number of GCSE grades a student has
    Dim count As Integer
    count = 0

    'sets the SQL statement
    sql = "SELECT DISTINCT Grade.GradeID, Student.Firstname, Student.Surname,
Subject.Subject, Grade.Grade, [Level].[Level], GradeType.GradeType, DatasetID.Year FROM
StudentGroup INNER JOIN Class ON StudentGroup.ClassID = Class.ClassID INNER JOIN Grade
INNER JOIN[Level] ON Grade.LevelID = [Level].LevelID INNER JOIN GradeType ON
Grade.GradeTypeID = GradeType.GradeTypeID INNER JOIN Student ON Grade.StudentID =
Student.StudentID ON StudentGroup.StudentID = Student.StudentID INNER JOIN Subject ON
Grade.SubjectID = Subject.SubjectID INNER JOIN DatasetID ON Grade.DatasetID =
DatasetID.DatasetID WHERE Student.StudentId = '" & StudentID & "' ORDER BY GradeID ASC"
    'the dataset is set after passing the following SQL statement and table into
the GenerateDataset sub of the public module
    ds = generateDataset(sql, "temp")

    'sets the datasource of the datagrid, the datamember and sets the datagrid as
visible
    Predictedgradesgrid.DataSource = ds
    Predictedgradesgrid.Visible = True
    Predictedgradesgrid.DataMember = "temp"
    'hides the GradeID column of the datagrid from the user
    Predictedgradesgrid.Columns("GradeID").Visible = False
    'sets the dt datatable as the dataset
    dt = ds.Tables(0)
    'automatically re-sizes the the columns of the datagrid
    Predictedgradesgrid.AutoResizeColumns()

    'this for next statement generates a total predicted value score by reading
the each row of the datatable and then applying a gcse weighting to the each grade
    For i = 0 To (dt.Rows.Count - 1)

        'if the value of the grade underneath the Grade header in the datatable on
the row of the i index is an A* then this if statement runs
        If dt.Rows(i)("Grade").ToString() = "A*" Then

```

```

'the GCSE gradeweighting variable is received after passing the
datatable and the row index into the GCSEGradeweighting function in the public module
GCSEgradeweighting = GCSEGradeweighting(i, dt)
'the predicted value is set as a national value but then is multiplied
by the weighting value
predictedvalue = 58 * GCSEgradeweighting
count = count + 1
End If

'all of the following if statements work like the one above for all the
other different grades
If dt.Rows(i)("Grade").ToString() = "A" Then
    GCSEgradeweighting = GCSEGradeweighting(i, dt)
    predictedvalue = 52 * GCSEgradeweighting
    count = count + 1
End If

If dt.Rows(i)("Grade").ToString() = "B" Then
    GCSEgradeweighting = GCSEGradeweighting(i, dt)
    predictedvalue = 46 * GCSEgradeweighting
    count = count + 1
End If

If dt.Rows(i)("Grade").ToString() = "C" Then
    GCSEgradeweighting = GCSEGradeweighting(i, dt)
    predictedvalue = 40 * GCSEgradeweighting
    count = count + 1
End If

If dt.Rows(i)("Grade").ToString() = "D" Then
    GCSEgradeweighting = GCSEGradeweighting(i, dt)
    predictedvalue = 34 * GCSEgradeweighting
    count = count + 1
End If

If dt.Rows(i)("Grade").ToString() = "E" Then
    GCSEgradeweighting = GCSEGradeweighting(i, dt)
    predictedvalue = 28 * GCSEgradeweighting
    count = count + 1
End If

If dt.Rows(i)("Grade").ToString() = "F" Then
    GCSEgradeweighting = GCSEGradeweighting(i, dt)
    predictedvalue = 22 * GCSEgradeweighting
    count = count + 1
End If

If dt.Rows(i)("Grade").ToString() = "4" Then
    GCSEgradeweighting = GCSEGradeweighting(i, dt)
    predictedvalue = 0
End If

If dt.Rows(i)("Grade").ToString() = "5" Then
    GCSEgradeweighting = GCSEGradeweighting(i, dt)
    predictedvalue = 0
End If

'the current predicted total equals the previous currentpredicted total +
the predicted value
currentpredictedtotal = currentpredictedtotal + predictedvalue
Next

'the average point score is set as a mean average by getting the total point
score and dividing by the number of datatable rows
averagepointscore = currentpredictedtotal / count

```

```

'creates a new variable that stores information for a button column that can
be added to the datagrid
Dim Columnview As New DataGridViewButtonColumn
With Columnview
    'sets the header text of the button column
    .HeaderText = "View Predicted Grades"
    'sets the name of the button column
    .Name = "Details"
    'sets the text displayed on the buttons of the button column in the
datagrid
    .Text = "Predicted Grades"
    .UseColumnTextForButtonValue = True
End With

'adds the button column to the datagrid
Predictedgradesgrid.Columns.Add(Columnview)
'auto resizes the column widths of the datagrid
End If
End Sub
'this sub is run when the Clear Selection button is clicked
Private Sub Clearselection_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Clearselection.Click
    'clears all controls on this form
    Me.Controls.Clear()
    're-initializes all components of this form
    InitializeComponent()
    'then re-loads this form with all components reset
    Predicted_Grades_Load(e, e)
End Sub
'this sub runs when all of the fields that require the user's input have been
completed
Private Sub formfieldgenerate()
    'selects the forms based on the user's choice of year
    sql = "SELECT Distinct Form FROM Student WHERE DatasetID = " &
Yearfield.SelectedIndex & ""
    'stores a SQL command that uses the SQL statement to query the database that is
found when following the connection string of con
    Using comm As SqlCommand = New SqlCommand(sql, con)
        'stores a SQL dataadareader which can read the contents of the sql query from
the database
        Dim rs As SqlDataReader = comm.ExecuteReader
        'stores a datatable
        Dim dt As DataTable = New DataTable
        'the datatable is filled with the values that the datareader reads
        dt.Load(rs)

        'sets the displaymemeber of the combobox
        formfield.DisplayMember = "Form"
        'sets the datasource of the combobox
        formfield.DataSource = dt
    End Using
End Sub
'this function handles every instance of the yearfield when it changes
Private Sub Yearfield_SelectedIndexChanged(sender As System.Object, e As
System.EventArgs) Handles Yearfield.SelectedIndexChanged
    'if the yearfield is not blank then the following objects become visible
    If Yearfield.Text <> "" Then
        formfieldgenerate()
        formfield.Visible = True
        labelform.Visible = True
        Searchgroup.Visible = True
    End If
End Sub

```

```

    Else
        'otherwise the following objects are not visible
        formfield.Visible = False
        labelform.Visible = False
        Searchgroup.Visible = False
    End If
End Sub
Private Sub Searchgroup_Click(sender As System.Object, e As System.EventArgs) Handles Searchgroup.Click

    Dim sql2 As String

    'sets the SQL statement
    sql2 = "SELECT DISTINCT Student.StudentID, Student.Firstname, Student.Surname FROM
StudentGroup INNER JOIN Class ON StudentGroup.ClassID = Class.ClassID INNER JOIN Grade
INNER JOIN[Level] ON Grade.LevelID = [Level].LevelID INNER JOIN GradeType ON
Grade.GradeTypeID = GradeType.GradeTypeID INNER JOIN Student ON Grade.StudentID =
Student.StudentID ON StudentGroup.StudentID = Student.StudentID INNER JOIN Subject ON
Grade.SubjectID = Subject.SubjectID INNER JOIN DatasetID ON Grade.DatasetID =
DatasetID.DatasetID WHERE Student.Form = '" & formfield.Text & "' AND Student.DatasetID =
'" & Yearfield.SelectedValue & "' ORDER BY Student.StudentID ASC"
        'passes the SQL2 statement into the grouppredictions sub which generates a
        datatable filled with predicted grades
        grouppredictions(sql2)

    End Sub
    Private Sub yearsearch_Click(sender As System.Object, e As System.EventArgs) Handles yearsearch.Click

        Dim sql2 As String
        'sets the SQL statement
        sql2 = "SELECT DISTINCT Student.StudentID, Student.Firstname, Student.Surname FROM
StudentGroup INNER JOIN Class ON StudentGroup.ClassID = Class.ClassID INNER JOIN Grade
INNER JOIN[Level] ON Grade.LevelID = [Level].LevelID INNER JOIN GradeType ON
Grade.GradeTypeID = GradeType.GradeTypeID INNER JOIN Student ON Grade.StudentID =
Student.StudentID ON StudentGroup.StudentID = Student.StudentID INNER JOIN Subject ON
Grade.SubjectID = Subject.SubjectID INNER JOIN DatasetID ON Grade.DatasetID =
DatasetID.DatasetID WHERE Student.DatasetID = '" & yearcombo.SelectedValue & "' ORDER BY
Student.StudentID ASC"
        grouppredictions(sql2)

    End Sub
    Private Sub Classsearch_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Classsearch.Click

        Dim sql2 As String
        'sets the SQL statement
        sql2 = "SELECT DISTINCT Student.StudentID, Student.Firstname, Student.Surname FROM
StudentGroup INNER JOIN Class ON StudentGroup.ClassID = Class.ClassID INNER JOIN Grade
INNER JOIN[Level] ON Grade.LevelID = [Level].LevelID INNER JOIN GradeType ON
Grade.GradeTypeID = GradeType.GradeTypeID INNER JOIN Student ON Grade.StudentID =
Student.StudentID ON StudentGroup.StudentID = Student.StudentID INNER JOIN Subject ON
Grade.SubjectID = Subject.SubjectID INNER JOIN DatasetID ON Grade.DatasetID =
DatasetID.DatasetID WHERE Class.Classgroup = '" & Classgroupfield.Text & "' ORDER BY
Student.StudentID ASC"
        grouppredictions(sql2)

    End Sub
    Sub grouppredictions(ByVal sql2 As String)
        grouppredictionsgrid.Visible = True
    End Sub

```

```

'calls the Loadgraphic class
Dim objPlsWait As New clsOPAWaitScreen
objPlsWait.ShowWaitScreen("")

'store subjects
Dim dt As New DataTable
'store students
Dim dt1 As New DataTable
'store studentsgrades
Dim dt2 As New DataTable
'store the grid that will be displayed
Dim dt3 As New DataTable
Dim ds As DataSet
'store index variables used in my for next statements
Dim i As Integer
Dim s As Integer
Dim z As Integer

'store the predictedvalue point score of a Grade
Dim predictedvalue As Double
'store the total of all the predictedvalues
Dim currentpredictedtotal As Double = 0
'store the actual predicted grade
Dim predictedgrade As String
'store a subject
Dim subject As String
'store a weighting variable that is applied to the predictedvalue depending on
the subject
Dim GCSEgradeweighting As Double

'Store a students firstname and surname
Dim firstname As String
Dim surname As String

'sets the other two grids on the form as non-visible
Predictedgradesgrid.Visible = False
Viewpredictionsgrid.Visible = False

'queries all of the subjects and places them in alphabetical order
sql = "SELECT DISTINCT Subject FROM Subject Order By Subject ASC"
'the dataset is set after passing the following SQL statement and table into the
GenerateDataset sub of the public module
ds = generateDataset(sql, "temp")
'the dt datatable is filled with the contents of the dataset
dt = ds.Tables(0)

'the dataset is set after passing the SQL2 statement which is set depending on the
user's filter selection, and table into the GenerateDataset sub of the public module
ds = generateDataset(sql2, "temp")
'the dt1 table is filled with a list of students
dt1 = ds.Tables(0)

'A firstname and surname column is added to the dt3 datatable which will be later
displayed to the user
dt3.Columns.Add("Firstname")
dt3.Columns.Add("Surname")

'this for next statement loops through all the rows in the students datatable and
stores the firstname and surname of a student
'these firstname and surnames are then written to the dt3 datatable
For i = 0 To (dt1.Rows.Count - 1)
    firstname = dt1.Rows(i)("Firstname").ToString()

```

```

surname = dt1.Rows(i)("Surname").ToString()
dt3.Rows.Add(firstname, surname)
Next

'This for next statement loops through the subjects datatable and sets a subject
variable for each row, this subject row variable is then added the dt3 datatable as a new
column heading
For i = 0 To (dt.Rows.Count - 1)
    subject = dt.Rows(i)("Subject").ToString()
    dt3.Columns.Add(subject)
Next

'this for next loop loops through every students
'for eachs student the dt2 datatable is filled with that students grades
For i = 0 To (dt1.Rows.Count - 1)
    averagepointscore = 0
    currentpredictedtotal = 0
    StudentID = dt1.Rows(i)("StudentID").ToString()
    sql = "SELECT DISTINCT Grade.GradeID, Student.Firstname, Student.Surname,
    Subject.Subject, Grade.Grade, [Level].[Level], GradeType.GradeType, DatasetID.Year FROM
    StudentGroup INNER JOIN Class ON StudentGroup.ClassID = Class.ClassID INNER JOIN Grade
    INNER JOIN[Level] ON Grade.LevelID = [Level].LevelID INNER JOIN GradeType ON
    Grade.GradeTypeID = GradeType.GradeTypeID INNER JOIN Student ON Grade.StudentID =
    Student.StudentID ON StudentGroup.StudentID = Student.StudentID INNER JOIN Subject ON
    Grade.SubjectID = Subject.SubjectID INNER JOIN DatasetID ON Grade.DatasetID =
    DatasetID.DatasetID WHERE Student.StudentId = '" & StudentID & "' ORDER BY GradeID ASC"
    ds = generateDataset(sql, "temp")
    dt2 = ds.Tables(0)

'counts the number of GCSE grades
Dim count As Integer
count = 0

'this for next statement then loops through that students grades and then
applies a predictedvalue depending on the grade, this grade later has a weighting applied
to it
For s = 0 To (dt2.Rows.Count - 1)

    'if the value of the grade underneath the Grade header in the datatable on
    the row of the i index is an A* then this if statement runs
    If dt2.Rows(s)("Grade").ToString() = "A*" Then
        'the GCSE gradeweighting variable is received after passing the
        datatable and the row index into the GCSEgradeweighting function in the public module
        GCSEgradeweighting = GCSEGradeweighting(s, dt2)
        'the predicted value is set as a national value but then is multiplied
        by the weighting value
        predictedvalue = 58 * GCSEgradeweighting
        count = count + 1
    End If
    'all of the following if statements work like the one above for all the
    other different grades
    If dt2.Rows(s)("Grade").ToString() = "A" Then
        GCSEgradeweighting = GCSEGradeweighting(s, dt2)
        predictedvalue = 52 * GCSEgradeweighting
        count = count + 1
    End If
    If dt2.Rows(s)("Grade").ToString() = "B" Then
        GCSEgradeweighting = GCSEGradeweighting(s, dt2)
        predictedvalue = 46 * GCSEgradeweighting
        count = count + 1
    End If
    If dt2.Rows(s)("Grade").ToString() = "C" Then

```

```

        GCSEgradeweighting = GCSEGradeweighting(s, dt2)
        predictedvalue = 40 * GCSEgradeweighting
        count = count + 1
    End If
    If dt2.Rows(s)("Grade").ToString() = "D" Then
        GCSEgradeweighting = GCSEGradeweighting(s, dt2)
        predictedvalue = 34 * GCSEgradeweighting
        count = count + 1
    End If
    If dt2.Rows(s)("Grade").ToString() = "E" Then
        GCSEgradeweighting = GCSEGradeweighting(s, dt2)
        predictedvalue = 28 * GCSEgradeweighting
        count = count + 1
    End If
    If dt2.Rows(s)("Grade").ToString() = "F" Then
        GCSEgradeweighting = GCSEGradeweighting(s, dt2)
        predictedvalue = 22 * GCSEgradeweighting
        count = count + 1
    End If
    If dt2.Rows(s)("Grade").ToString() = "4" Then
        GCSEgradeweighting = GCSEGradeweighting(s, dt2)
        predictedvalue = 0
    End If
    If dt2.Rows(s)("Grade").ToString() = "5" Then
        GCSEgradeweighting = GCSEGradeweighting(s, dt2)
        predictedvalue = 0
    End If
    'the current predicted total equals the previous currentpredicted total +
    the predicted value
    currentpredictedtotal = currentpredictedtotal + predictedvalue
Next

'a mean average point score for the student is created by taking the total of
all the predicted values and dividing by the number of grades
averagepointscore = currentpredictedtotal / count

'this for next statement then loops through every subject within the dt
datatable
For z = 0 To (dt.Rows.Count - 1)
    'the subject variable is set as the value of the cell that is part of the
    row index z
    subject = dt.Rows(z)("Subject").ToString()
    'the actual predicted grade is set by passing the subject, row index,
    datatable and average point score to the A2predictedgrade sub in the public module
    predictedgrade = A2predictedgrade(subject, z, dt, averagepointscore)
    'the predicted grade is then written on the row index of i underneath the
    subject header variable
    dt3.Rows(i)(subject) = predictedgrade
Next
Next
'the grids datasource is set to dt3
grouppredictionsgrid.DataSource = dt3
'the grid auto-resizes its columns
grouppredictionsgrid.AutoResizeColumns()
'the button that creates a spreadsheet file becomes visible to the user
createcsv.Visible = True

'closes the Laod graphic form and resets its objplswait variable
objPlsWait.CloseWaitScreen()
objPlsWait = Nothing

End Sub

```

```

Private Sub btnExport_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles createcsv.Click

    'Build the CSV file data as a Comma separated string.
    Dim csv As String = String.Empty

    'Adds the Header row for CSV file by taking all the headers of each datagrid
    'column.
    For Each column As DataGridViewColumn In grouppredictionsgrid.Columns
        csv += column.HeaderText & ","c
    Next

    'Adds a new line
    csv += vbCr & vbLf

    'Deals with adding the Rows to the string based on tyhe values of the rows in the
    'datagrid
    For Each row As DataGridViewRow In grouppredictionsgrid.Rows
        For Each cell As DataGridViewCell In row.Cells
            'Adds the Data rows.
            csv += cell.Value.ToString().Replace(", ", ";") & ","
        Next

        'Adds new line.
        csv += vbCr & vbLf
    Next

    'stores a save dialog variable
    Dim saveFileDialog1 As New SaveFileDialog()

    'sets the file formats that the savedialog is restricted to, in this case a CSV
    'file
    saveFileDialog1.Filter = "Spreadsheet CSV file (*.csv)|*.csv"
    'Sets the filter index of the save dialog as the user is restricted to saving the
    'file as a CSV
    saveFileDialog1.FilterIndex = 1
    saveFileDialog1.RestoreDirectory = True

    'if the user clicks the save option within the save dialog box the CSV file is
    'saved in the current directory that is shown in the savedialog.
    If saveFileDialog1.ShowDialog() = DialogResult.OK Then
        'Creates a streamwriter that creates a file with the user's savedialog name
        'for the file
        Dim sw As StreamWriter = New StreamWriter(saveFileDialog1.OpenFile())
        'if the stream writer is not empty then the streamwriter saves a CSV formatted
        'file with the CSV string that has been created
        If (sw IsNot Nothing) Then
            sw.WriteLine(csv)
            'closes the streamwriter
            sw.Close()
        End If
    End If
End Sub

'deals with the print form button
Private Sub printbutton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles printbutton.Click
    'sets the printer settings to default
    PrintForm.PrinterSettings = PrintDialog.PrinterSettings
    If PrintDialog.ShowDialog() = DialogResult.OK Then
        'allows the user to change print settings
        PrintForm.PrinterSettings = PrintDialog.PrinterSettings
        'changes the orientation of the printed document to landscape
    End If
End Sub

```

```

        Me.PrintForm.PrinterSettings.DefaultPageSettings.Landscape = True
        'refreshes the form so that the printdialog is not printed along with the form
        Me.Refresh()
        'prints the current form
        Me.PrintForm.Print()
        'Percentageschart.Printing.Print(True)
    End If
End Sub
'this sub handles every instance of the yearcombo field when it is altered
Private Sub yearcombo_SelectedIndexChanged(sender As System.Object, e As
System.EventArgs) Handles yearcombo.SelectedIndexChanged
    'if the yearcombo is not blank then the search button appears
    If yearcombo.Text <> "" Then
        yearsearch.Visible = True
    Else
        yearsearch.Visible = False
    End If
End Sub

'this sub runs when the text within the subject field is altered
Private Sub SubjectField_SelectedIndexChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles SubjectField.SelectedIndexChanged
    If SubjectField.Text = "" Then
        Classlabel.Visible = False
        Classsearch.Visible = False
        Classgroupfield.Visible = False
    ElseIf classyearfield.Text <> "" And SubjectField.Text <> "" Then
        Classfieldgenerate()
    End If
End Sub

'this sub runs when all of the fields that require the user's input have been
completed
Private Sub Classfieldgenerate()
    'selects the classes based on the user's choice of subject and year
    sql = "SELECT Distinct Class.ClassID, Class.ClassGroup FROM Department INNER JOIN
Subject ON Department.DepartmentID = Subject.DepartmentID INNER JOIN DatasetID INNER JOIN
Class ON DatasetID.DatasetID = Class.DatasetID ON Subject.SubjectID = Class.SubjectID
INNER JOIN StudentGroup ON Class.ClassID = StudentGroup.ClassID WHERE Class.SubjectID = ''
& SubjectField.SelectedValue & '' And Class.DatasetID = '' & classyearfield.SelectedValue
& '' ORDER BY Class.ClassID ASC"
    'stores a SQL command that uses the SQL statement to query the database that is
found when following the connection string of con
    Using comm As SqlCommand = New SqlCommand(sql, con)
        'stores a SQL dataadareader which can read the contents of the sql query from
the database
        Dim rs As SqlDataReader = comm.ExecuteReader
        'stores a datatable
        Dim dt As DataTable = New DataTable
        'the datatable is filled with the values that the datareader reads
        dt.Load(rs)

        'the index of each item(class) in the combobox is set as its correspoinding ID
        'in the database
        Classgroupfield.ValueMember = "ClassID"
        'sets the displaymemeber of the combobox
        Classgroupfield.DisplayMember = "ClassGroup"
        'sets the datasource of the combobox
        Classgroupfield.DataSource = dt
    End Using
    'makes the class combobox and its label visible to the user

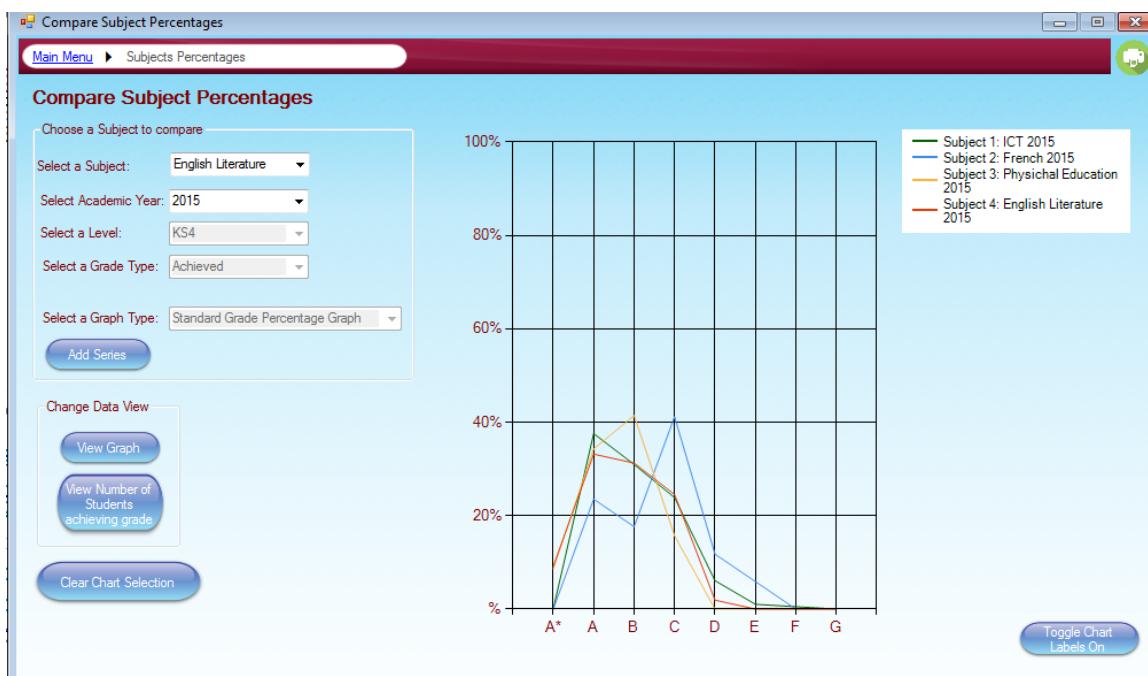
```

```
Classlabel.Visible = True
Classgroupfield.Visible = True
End Sub

'this sub handles the instance the classyearfield is changed
Private Sub classyearfield_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles classyearfield.SelectedIndexChanged
    'sets the visible property of the search button depending on whether there is text
    in the classyearfield
    If classyearfield.Text = "" Then
        Classlabel.Visible = False
        Classsearch.Visible = False
        Classgroupfield.Visible = False
    ElseIf classyearfield.Text <> "" And SubjectField.Text <> "" Then
        'runs the classfieldgenerate sub
        Classfieldgenerate()
    End If
End Sub
'this sub handles the instance the classgroupfield is changed
Private Sub Classgroupfield_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Classgroupfield.SelectedIndexChanged
    'sets the visible property of the search button depending on whether there is text
    in the classgroupfield
    If Classgroupfield.Text <> "" Then
        Classsearch.Visible = True
    Else
        Classsearch.Visible = False
    End If
End Sub
End Class
```

Compare Subject Percentages

This form allows a user to compare grade statistics of achieved GCSE grades or predicted A2 grades. This is done by comparing individual subjects within years, a user can select which subject they want to compare and which grade type and level they want to compare. For example a user could compare the achieved GCSE English literature statistics for up to four consecutive years. A maximum of four subjects can be compared at any one time. The statistics can be displayed as percentages on a linear line graph or a cumulative grade line graph. There is also a viewing mode that allows the user to view the raw statistics of students achieving or predicted individual grades and the number of students achieving or predicted A* - C grades. At any time the current subjects that are being compared can be cleared so that a new set of subjects can be compared. A printed copy of this form can be produced by using the print icon, which is very useful as a copy of the raw statistics can be produced or a copy of the graph comparisons can be produced.



The screenshot shows the same application window with the "Subjects Percentages" tab selected. It displays four tables of raw achievement data:

- Subject 1: ICT 2015**

Grade	Number of Students achieving Grade
A*	0 out of 197
A	74 out of 197
B	61 out of 197
C	47 out of 197
D	12 out of 197
E	2 out of 197
F	1 out of 197
G	0 out of 197
A*-C	182 out of 197 (~92%)
- Subject 2: French 2015**

Grade	Number of Students achieving Grade
A*	0 out of 17
A	4 out of 17
B	3 out of 17
C	7 out of 17
D	2 out of 17
E	1 out of 17
F	0 out of 17
G	0 out of 17
A*-C	14 out of 17 (~82%)
- Subject 3: Physical Education 2015**

Grade	Number of Students achieving Grade
A*	6 out of 70
A	24 out of 70
B	29 out of 70
C	11 out of 70
D	0 out of 70
E	0 out of 70
F	0 out of 70
G	0 out of 70
A*-C	70 out of 70 (~100%)
- Subject 4: English Literature 2015**

Grade	Number of Students achieving Grade
A*	19 out of 208
A	69 out of 208
B	65 out of 208
C	51 out of 208
D	4 out of 208
E	0 out of 208
F	0 out of 208
G	0 out of 208
A*-C	204 out of 208 (~98%)

On the left, there is a "Change Data View" section with "View Graph" and "View Number of Students achieving grade" buttons, and a "Clear Chart Selection" button. On the right, there is a "Toggle Chart Labels On" button.

Addsearch System.Windows.Forms.Button
banner System.Windows.Forms.PictureBox
chartlabel System.Windows.Forms.Label
Clearselection System.Windows.Forms.Button
Compare_Subject_Percentages System.Windows.Forms.Form
Comparesubjectgroup System.Windows.Forms.GroupBox
Gradetypefield System.Windows.Forms.ComboBox
Graphtypefield System.Windows.Forms.ComboBox
Graphview System.Windows.Forms.Button
Groupview System.Windows.Forms.GroupBox
labelstoggle System.Windows.Forms.Button
Levelfield System.Windows.Forms.ComboBox
Levellabel System.Windows.Forms.Label
MainMenu System.Windows.Forms.LinkLabel
printbutton System.Windows.Forms.Button
PrintDialog System.Windows.Forms.PrintDialog
PrintForm Microsoft.VisualBasic.PowerPacks.Printing.PrintForm
Seriestable1 System.Windows.Forms.TableLayoutPanel
Seriestable2 System.Windows.Forms.TableLayoutPanel
Seriestable3 System.Windows.Forms.TableLayoutPanel
Seriestable4 System.Windows.Forms.TableLayoutPanel
SubjectField System.Windows.Forms.ComboBox
subjectlabel System.Windows.Forms.Label
Subjectpercentageschart System.Windows.Forms.DataVisualization.Charting.Chart
subjectpercentageslink System.Windows.Forms.LinkLabel
Table1label System.Windows.Forms.Label
Table2label System.Windows.Forms.Label
Table3label System.Windows.Forms.Label
Table4label System.Windows.Forms.Label
TablesView System.Windows.Forms.Button
tabletitle1 System.Windows.Forms.Label
tabletitle2 System.Windows.Forms.Label
tabletitle3 System.Windows.Forms.Label
tabletitle4 System.Windows.Forms.Label
tbl1a System.Windows.Forms.Label
tbl1astar System.Windows.Forms.Label
tbl1atoc System.Windows.Forms.Label
tbl1b System.Windows.Forms.Label
tbl1c System.Windows.Forms.Label
tbl1d System.Windows.Forms.Label

tbl1e System.Windows.Forms.Label
tbl1f System.Windows.Forms.Label
tbl2a System.Windows.Forms.Label
tbl2astar System.Windows.Forms.Label
tbl2atoc System.Windows.Forms.Label
tbl2b System.Windows.Forms.Label
tbl2c System.Windows.Forms.Label
tbl2d System.Windows.Forms.Label
tbl2e System.Windows.Forms.Label
tbl2f System.Windows.Forms.Label
tbl3a System.Windows.Forms.Label
tbl3astar System.Windows.Forms.Label
tbl3atoc System.Windows.Forms.Label
tbl3b System.Windows.Forms.Label
tbl3c System.Windows.Forms.Label
tbl3d System.Windows.Forms.Label
tbl3e System.Windows.Forms.Label
tbl3f System.Windows.Forms.Label
tbl4a System.Windows.Forms.Label
tbl4astar System.Windows.Forms.Label
tbl4atoc System.Windows.Forms.Label
tbl4b System.Windows.Forms.Label
tbl4c System.Windows.Forms.Label
tbl4d System.Windows.Forms.Label
tbl4e System.Windows.Forms.Label
tbl4f System.Windows.Forms.Label
Yearfield System.Windows.Forms.ComboBox
yearlabel System.Windows.Forms.Label

Parameter Name	Description
CompareSubject_Percentages	This sub connects to the database server when the form initially loads and dynamically generates the list properties of all the combo boxes on the form and sets their value members. It also sets the default text property of the graph type combo box.
MainMenu_LinkClicked	When the main menu link label is clicked all open forms close and the main menu displays.
textvalidate	This function sets a true or false condition depending on whether any textboxes on the form are blank. It also sets the background colour of textboxes appropriately to indicate to the user that fields have not been completed.
KS4percentages	Selects all grades within the database and then counts the number of each KS4 grade e.g. A's B's. It then sets the text properties of the values within the statistics table and adds data points to the series of the chart, these data points depend on the user's selection of a standard or cumulative graph type. It then adds the corresponding data point axis labels for each KS4 grade.
KS2percentages	Selects all grades within the database and then counts the number of each KS2 grade e.g. 3's 4's. It then sets the text properties of the values within the statistics table and adds data points to the series of the chart, these data points depend on the user's selection of a standard or cumulative graph type. It then adds the corresponding data point axis labels for each KS2 grade.
A2Predictions	This sub generates a series of predicted grades by first selecting all students that match the filter criteria, and calculating an average point score for each student based on their GCSE grades. From this average point score a predicted grade is then generated for the selected subject the user want to view predicted grades for. Once all of the predicted grades have been generated, data points are added to the chart series, these are either linear data points or cumulative data points depending on the users graph choice. The label properties of the statistics table are also set as the number of each predicted grade. Corresponding data point axis labels for each predicted grade are also added to the chart.

Addsearch_Click	When the add series button is clicked by the user, this sub determines which type of series to add to the chart. It is also responsible for validation and formatting the chart.
Clearselection_Click	Creates a new instance of every object on the form (essentially re-loads the form).
labelstoggle_Click	Handles the button click event of the Toggle chart labels button. When clicked an index variable loops through all series and changes the visible property of the labels to false. It also changes the text property of the button.
Graphview_Click	When the view graph button is clicked all objects associated with the graph visible to the user and all objects associated with the statistics tables non-visible to the user.
TablesView_Click	When the view table's button is clicked all objects associated with the graph become non-visible to the user and all objects associated with the statistics tables become visible to the user.
printbutton_Click	This sub allows the user to print a hard copy of the form when the print icon is clicked, additional print dialog options are then displayed allowing the user to choose printer and number of copies amongst other settings.
Levelefield_SelectedIndexChanged	Whenever the text value of the level field is changed this sub determines whether the grade type field is either enabled or not enabled.

```

'Imports the following system code commands so that I can use them appropriately
Imports System.Data.SqlClient
Imports System.Data
Imports System.Configuration
Imports System.Data.DataTable
Public Class Compare_Subject_Percentages
    'stores series as an empty string that can be used within this class
    Dim series As String = Nothing
    'i and p are used as an index variable in my for next statements
    Dim i As Integer = 0
    Dim p As Integer = 0
    'these variables store the number of grades counted
    Dim threegrade As Integer = 0
    Dim fourgrade As Integer = 0
    Dim fivegrade As Integer = 0
    Dim Astargrade As Integer
    Dim Agrade As Integer = 0
    Dim Bgrade As Integer = 0
    Dim Cgrade As Integer = 0
    Dim Dgrade As Integer = 0
    Dim Egrade As Integer = 0
    Dim Fgrade As Integer = 0
    Dim Ggrade As Integer = 0
    'this sub runs when the form initially loads
    Private Sub CompareSubject_Percentages(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Me.Load
        'runs the databaseconnect sub from the public module
        DatabaseConnect()
        'stores dt as datatable
        Dim dt As DataTable
        p = 0

        'Uses the Generate combo function of the module to populate a combobox by passing
        the table, field and needempty variables to the Generatecombo function and then returning
        the result in a datatable
        dt = GenerateCombo("Subject", "SubjectID, Subject", True)
        'sets the datasource of the combobox
        SubjectField.DataSource = dt
        'sets the display member of the combobox
        SubjectField.DisplayMember = "Subject"
        'the index of each item in the combobox is set as their corresponding ID from the
        database
        SubjectField.ValueMember = "SubjectID"

        dt = GenerateCombo("DatasetID", "DatasetID, Year", True)
        Yearfield.DataSource = dt
        Yearfield.DisplayMember = "Year"
        Yearfield.ValueMember = "DatasetID"

        dt = GenerateCombo("Level", "levelID, Level", True)
        Levelfield.DataSource = dt
        Levelfield.DisplayMember = "Level"
        Levelfield.ValueMember = "LevelID"

        dt = GenerateCombo("GradeType", "GradeTypeID, GradeType", True)
        Gradetypefield.DataSource = dt
        Gradetypefield.DisplayMember = "GradeType"
        Gradetypefield.ValueMember = "GradeTypeID"

        'sets the default text of the graphytypefield
        Graphtypefield.Text = "Standard Grade Percentage Graph"
    End Sub

```

```

'this sub handles the instance the main menu link is clicked by the user
Private Sub MainMenu_LinkClicked(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles MainMenu.LinkClicked
    'closes this form and shows the main menu form
    Main_Menu.Show()
    Me.Close()
End Sub
'this function returns a true or false state depending on whether the conditions
within this sub have been met or not
Public Function textvalidate() As Boolean
    'initially the functions state is set to true
    textvalidate = True
    'if any of the textboxes are left blank then the function returns a false state
    'and the background of the textbox field changes to red, otherwise the background of the
    'textbox changes to white
    If Levelfield.Text = "" Then
        textvalidate = False
        Levelfield.BackColor = Color.Salmon
    Else
        Levelfield.BackColor = Color.White
    End If
    If SubjectField.Text = "" Then
        textvalidate = False
        SubjectField.BackColor = Color.Salmon
    Else
        SubjectField.BackColor = Color.White
    End If
    If Gradetypefield.Text = "" Then
        textvalidate = False
        Gradetypefield.BackColor = Color.Salmon
    Else
        Gradetypefield.BackColor = Color.White
    End If
    If Yearfield.Text = "" Then
        textvalidate = False
        Yearfield.BackColor = Color.Salmon
    Else
        Yearfield.BackColor = Color.White
    End If
End Function
'this sub deals with KS4 percentages
Public Sub kS4percentages()
    'stores ds as a dataset
    Dim ds As New DataSet
    'stores dt as a datatable
    Dim dt As New DataTable

    ' This SQL statement fills a dataset depending on the users department choice
    sql = "SELECT [A*],[A],[B],[C],[D],[E],[F],[G] FROM Grade PIVOT (Count(GradeID)
FOR Grade IN ([A*],[A],[B],[C],[D],[E],[F],[G])) AS PVTTable WHERE SubjectID = '" &
    SubjectField.SelectedValue & "' and DatasetID = '" & Yearfield.SelectedValue & "' and
    LevelID = '" & Levelfield.SelectedValue & "' and GradeTypeID = '" &
    Gradetypefield.SelectedValue & "'"
    'the dataset is set after passing the following SQL statement and table into the
    GenerateDataset sub of the public module
    ds = generateDataset(sql, "temp")
    'the datatable contents is set as the returned dataset once the SQL statement has
    filled the dataset
    dt = ds.Tables(0)

    'this for next statement loops itself to the last row containing a grade
    'it counts the number of each grade within the dataset

```

```

For i = 0 To (dt.Rows.Count - 1)
    'if there is a matching grade under any of the columns in the dataset for each
    'row then the count of that corresponding grade count increases by 1
    If dt.Rows(i)("A*").ToString() = "1" Then
        Astargrade = Astargrade + 1
    End If
    If dt.Rows(i)("A").ToString() = "1" Then
        Agrade = Agrade + 1
    End If
    If dt.Rows(i)("B").ToString() = "1" Then
        Bgrade = Bgrade + 1
    End If
    If dt.Rows(i)("C").ToString() = "1" Then
        Cgrade = Cgrade + 1
    End If
    If dt.Rows(i)("D").ToString() = "1" Then
        Dgrade = Dgrade + 1
    End If
    If dt.Rows(i)("E").ToString() = "1" Then
        Egrade = Egrade + 1
    End If
    If dt.Rows(i)("F").ToString() = "1" Then
        Fgrade = Fgrade + 1
    End If
    If dt.Rows(i)("G").ToString() = "1" Then
        Ggrade = Ggrade + 1
    End If
Next

'sets the value of the text labels within the table that corresponds to the index
value of p
If p = 0 Then
    'the text values of the labels in the table when the tableviewbutton is
    clicked are set to their corresponding number of grades counted from the dataset
    tbl1astar.Text = Astargrade & " out of " & dt.Rows.Count
    tbl1a.Text = Agrade & " out of " & dt.Rows.Count
    tbl1b.Text = Bgrade & " out of " & dt.Rows.Count
    tbl1c.Text = Cgrade & " out of " & dt.Rows.Count
    tbl1d.Text = Dgrade & " out of " & dt.Rows.Count
    tbl1e.Text = Egrade & " out of " & dt.Rows.Count
    tbl1f.Text = Fgrade & " out of " & dt.Rows.Count
    tbl1ggrade.Text = Ggrade & " out of " & dt.Rows.Count
    'this sets the label of the table, so the user knows what the data in the
    table correlates to
    Table1label.Text = "Subject " & p + 1 & ":" & SubjectField.Text & " " &
Yearfield.Text
    'this sets the atoc label text of the table by adding up all the grades from
    A* to c and then dividing by the total number of grades in the dataset, which is then
    converted to a percentage and displayed as a percentage
    tbl1atoc.Text = (Agrade + Bgrade + Cgrade + Astargrade) & " out of " &
dt.Rows.Count & " (~" & Int(((Agrade + Bgrade + Cgrade + Astargrade) / dt.Rows.Count) *
100) & "%)"
End If
If p = 1 Then
    tbl2astar.Text = Astargrade & " out of " & dt.Rows.Count
    tbl2a.Text = Agrade & " out of " & dt.Rows.Count
    tbl2b.Text = Bgrade & " out of " & dt.Rows.Count
    tbl2c.Text = Cgrade & " out of " & dt.Rows.Count
    tbl2d.Text = Dgrade & " out of " & dt.Rows.Count
    tbl2e.Text = Egrade & " out of " & dt.Rows.Count
    tbl2f.Text = Fgrade & " out of " & dt.Rows.Count
    tbl2ggrade.Text = Ggrade & " out of " & dt.Rows.Count

```

```

Table2label.Text = "Subject " & p + 1 & ":" & SubjectField.Text & " " &
Yearfield.Text
tbl2atoc.Text = (Agrade + Bgrade + Cgrade + Astargrade) & " out of " &
dt.Rows.Count & " (~" & Int(((Agrade + Bgrade + Cgrade + Astargrade) / dt.Rows.Count) *
100) & "%)"
End If
If p = 2 Then
    tbl3astar.Text = Astargrade & " out of " & dt.Rows.Count
    tbl3a.Text = Agrade & " out of " & dt.Rows.Count
    tbl3b.Text = Bgrade & " out of " & dt.Rows.Count
    tbl3c.Text = Cgrade & " out of " & dt.Rows.Count
    tbl3d.Text = Dgrade & " out of " & dt.Rows.Count
    tbl3e.Text = Egrade & " out of " & dt.Rows.Count
    tbl3f.Text = Fgrade & " out of " & dt.Rows.Count
    tbl3ggrade.Text = Ggrade & " out of " & dt.Rows.Count
    Table3label.Text = "Subject " & p + 1 & ":" & SubjectField.Text & " " &
Yearfield.Text
    tbl3atoc.Text = (Agrade + Bgrade + Cgrade + Astargrade) & " out of " &
dt.Rows.Count & " (~" & Int(((Agrade + Bgrade + Cgrade + Astargrade) / dt.Rows.Count) *
100) & "%)"
End If
If p = 3 Then
    tbl4astar.Text = Astargrade & " out of " & dt.Rows.Count
    tbl4a.Text = Agrade & " out of " & dt.Rows.Count
    tbl4b.Text = Bgrade & " out of " & dt.Rows.Count
    tbl4c.Text = Cgrade & " out of " & dt.Rows.Count
    tbl4d.Text = Dgrade & " out of " & dt.Rows.Count
    tbl4e.Text = Egrade & " out of " & dt.Rows.Count
    tbl4f.Text = Fgrade & " out of " & dt.Rows.Count
    tbl4ggrade.Text = Ggrade & " out of " & dt.Rows.Count
    Table4label.Text = "Subject " & p + 1 & ":" & SubjectField.Text & " " &
Yearfield.Text
    tbl4atoc.Text = (Agrade + Bgrade + Cgrade + Astargrade) & " out of " &
dt.Rows.Count & " (~" & Int(((Agrade + Bgrade + Cgrade + Astargrade) / dt.Rows.Count) *
100) & "%)"
End If

'this if statement selects what type of graph to display based on the user's
selection
If Graphtypefield.Text = "Cumulative Grade Percentage Graph" Then
    'name of the series is set dynamically depending on users choices
    series = "Subject " & p + 1 & ":" & SubjectField.Text & " " & Yearfield.Text
    'adds a series to the graph
    Subjectpercentageschart.Series.Add(series)
    'adds datapoints to the series of the graph, these points work cumulatively so
    that each following data point adds all the previous data points values to it
    Subjectpercentageschart.Series(series).Points.AddY(Astargrade / dt.Rows.Count)
    Subjectpercentageschart.Series(series).Points.AddY((Agrade + Astargrade) /
    dt.Rows.Count)
    Subjectpercentageschart.Series(series).Points.AddY((Bgrade + Astargrade +
    Agrade) / dt.Rows.Count)
    Subjectpercentageschart.Series(series).Points.AddY((Cgrade + Astargrade +
    Agrade + Bgrade) / dt.Rows.Count)
    Subjectpercentageschart.Series(series).Points.AddY((Dgrade + Astargrade +
    Agrade + Bgrade + Cgrade) / dt.Rows.Count)
    Subjectpercentageschart.Series(series).Points.AddY((Egrade + Dgrade +
    Astargrade + Agrade + Bgrade + Cgrade) / dt.Rows.Count)
    Subjectpercentageschart.Series(series).Points.AddY((Fgrade + Egrade + Dgrade +
    Astargrade + Agrade + Bgrade + Cgrade) / dt.Rows.Count)
    Subjectpercentageschart.Series(series).Points.AddY((Ggrade + Fgrade + Egrade +
    Dgrade + Astargrade + Agrade + Bgrade + Cgrade) / dt.Rows.Count)
Else

```

```

    'name of the series is set dynamically depending on users choices
    series = "Subject " & p + 1 & ":" & SubjectField.Text & " " & Yearfield.Text
    'adds a series to the graph
    Subjectpercentageschart.Series.Add(series)
    'adds datapoints to the series of the graph equivalent to the number of grades
    counted
        Subjectpercentageschart.Series(series).Points.AddY(Astargrade / dt.Rows.Count)
        Subjectpercentageschart.Series(series).Points.AddY(Agrade / dt.Rows.Count)
        Subjectpercentageschart.Series(series).Points.AddY(Bgrade / dt.Rows.Count)
        Subjectpercentageschart.Series(series).Points.AddY(Cgrade / dt.Rows.Count)
        Subjectpercentageschart.Series(series).Points.AddY(Dgrade / dt.Rows.Count)
        Subjectpercentageschart.Series(series).Points.AddY(Egrade / dt.Rows.Count)
        Subjectpercentageschart.Series(series).Points.AddY(Fgrade / dt.Rows.Count)
        Subjectpercentageschart.Series(series).Points.AddY(Ggrade / dt.Rows.Count)
    End If

    'sets the labels of the datapoints on the series
    Subjectpercentageschart.Series(series).Points(0).AxisLabel = "A*"
    Subjectpercentageschart.Series(series).Points(1).AxisLabel = "A"
    Subjectpercentageschart.Series(series).Points(2).AxisLabel = "B"
    Subjectpercentageschart.Series(series).Points(3).AxisLabel = "C"
    Subjectpercentageschart.Series(series).Points(4).AxisLabel = "D"
    Subjectpercentageschart.Series(series).Points(5).AxisLabel = "E"
    Subjectpercentageschart.Series(series).Points(6).AxisLabel = "F"
    Subjectpercentageschart.Series(series).Points(7).AxisLabel = "G"
End Sub
'this sub deals with KS2 percentages and works similarly to the Sub above
Public Sub KS2percentages()
    Dim ds As New DataSet
    Dim dt As New DataTable

    ' This SQL statement fills a dataset depending on the users class choice
    sql = "SELECT [3],[4],[5] FROM Grade PIVOT (Count(GradeID) FOR Grade IN
([3],[4],[5])) AS PVTTable WHERE SubjectID = '" & SubjectField.SelectedValue & "' and
DatasetID = '" & Yearfield.SelectedValue & "' and LevelID = '" & Levelfield.SelectedValue
& "' and GradeTypeID = '" & Gradetypefield.SelectedValue & "'"
    ds = generateDataset(sql, "temp")
    dt = ds.Tables(0)

    'counts the number of KS2 grades
    For i = 0 To (dt.Rows.Count - 1)
        If dt.Rows(i)("3").ToString() = "1" Then
            threegrade = threegrade + 1
        End If
        If dt.Rows(i)("4").ToString() = "1" Then
            fourgrade = fourgrade + 1
        End If
        If dt.Rows(i)("5").ToString() = "1" Then
            fivegrade = fivegrade + 1
        End If
    Next

    'sets the type of series to add to the graph based on whether the user wants to
    see a cumulative or non-cumulative graph
    If Graphtypefield.Text = "Cumulative Grade Percentage Graph" Then
        series = "Subject " & p + 1 & ":" & SubjectField.Text & " " & Yearfield.Text
        Subjectpercentageschart.Series.Add(series)
        'adds a cumulative series of datapoints
        Subjectpercentageschart.Series(series).Points.AddY(((fivegrade) /
        dt.Rows.Count))
        Subjectpercentageschart.Series(series).Points.AddY(((fourgrade + fivegrade) /
        dt.Rows.Count)))
    End If

```

```

        Subjectpercentageschart.Series(series).Points.AddY(((threegrade + fourgrade +
fivegrade) / dt.Rows.Count)))
    Else
        series = "Subject " & p + 1 & ":" & SubjectField.Text & " " & Yearfield.Text
        Subjectpercentageschart.Series.Add(series)
        'adds a non-cumulative series of datapoints
        Subjectpercentageschart.Series(series).Points.AddY(((fivegrade) /
dt.Rows.Count))
        Subjectpercentageschart.Series(series).Points.AddY(((fourgrade) /
dt.Rows.Count))
        Subjectpercentageschart.Series(series).Points.AddY(((threegrade) /
dt.Rows.Count))
    End If

    Subjectpercentageschart.Series(series).Points(0).AxisLabel = "3"
    Subjectpercentageschart.Series(series).Points(1).AxisLabel = "4"
    Subjectpercentageschart.Series(series).Points(2).AxisLabel = "5"
End Sub
'this sub deals with A2 prediction percentages
Public Sub A2Predictions(ByVal sql2 As String)
    'stores subjects
    Dim dt As New DataTable
    'stores students
    Dim dt1 As New DataTable
    'stores studentsgrades
    Dim dt2 As New DataTable
    'stores the grid that will be displayed
    Dim dt3 As New DataTable
    Dim ds As DataSet
    'stores index variables used in my for next statements
    Dim i As Integer
    Dim s As Integer

    'stores the predictedvalue point score of a Grade
    Dim predictedvalue As Double
    'stores the total of all the predictedvalues
    Dim currentpredictedtotal As Double = 0
    'stores the actual predicted grade
    Dim predictedgrade As String
    'stores a subject
    Dim subject As String
    'stores a weighting variable that is applied to the predictedvalue depending on
the subject
    Dim GCSEgradeweighting As Double
    'stores the averagepointscore for a student
    Dim averagepointscore As Double
    'stores the ID of a student
    Dim StudentID As Integer

    'queries all of the subjects and places them in alphabetical order
    sql = "SELECT DISTINCT Subject FROM Subject Where Subject.Subject = '" &
SubjectField.Text & "' ORDER BY Subject ASC"
    'the dataset is set after passing the following SQL statement and table into the
GenerateDataset sub of the public module
    ds = generateDataset(sql, "temp")
    'the dt datatable is filled with the a list of subjects from the database
    dt = ds.Tables(0)

    'the dataset is set after passing the SQL2 statement which is set depending on the
user's filter selection, and table into the GenerateDataset sub of the public module
    ds = generateDataset(sql2, "temp")
    'the dt1 table is filled with a list of students

```

```

dt1 = ds.Tables(0)

'this for next loop loops through every student
'for eachs student the dt2 datatable is filled with that students grades
dt3.Columns.Add("Grade")
For i = 0 To (dt1.Rows.Count - 1)
    averagepointscore = 0
    currentpredictedtotal = 0
    StudentID = dt1.Rows(i)("StudentID").ToString()
    sql = "SELECT DISTINCT Grade.GradeID, Student.Firstname, Student.Surname,
Subject.Subject, Grade.Grade, [Level].[Level], GradeType.GradeType, DatasetID.Year FROM
StudentGroup INNER JOIN Class ON StudentGroup.ClassID = Class.ClassID INNER JOIN Grade
INNER JOIN[Level] ON Grade.LevelID = [Level].LevelID INNER JOIN GradeType ON
Grade.GradeTypeID = GradeType.GradeTypeID INNER JOIN Student ON Grade.StudentID =
Student.StudentID ON StudentGroup.StudentID = Student.StudentID INNER JOIN Subject ON
Grade.SubjectID = Subject.SubjectID INNER JOIN DatasetID ON Grade.DatasetID =
DatasetID.DatasetID WHERE Student.StudentId = '" & StudentID & "' ORDER BY GradeID ASC"
    ds = generateDataset(sql, "temp")
    dt2 = ds.Tables(0)

'this variable is used to count the number of GCSE grades a student has
Dim count As Integer
count = 0

'this for next statement then loops through all of a students grades and then
applies a predictedvalue depending on the grade, this grade later has a weighting applied
to it
    For s = 0 To (dt2.Rows.Count - 1)

        'if the value of the grade underneath the Grade header in the datatable on
        the row of the i index is an A* then this if statement runs
        If dt2.Rows(s)("Grade").ToString() = "A*" Then
            'the GCSE gradeweighting variable is recieved after passing the
            datatable and the row index into the GCSEgradeweighting function in the public module
            GCSEgradeweighting = GCSEGradeweighting(s, dt2)
            'the predicted value is set as a national value but then is multiplied
            by the weighting value
            predictedvalue = 58 * GCSEgradeweighting
            count = count + 1
        End If
        'all of the following if statements work like the one above for all the
        other different grades
        If dt2.Rows(s)("Grade").ToString() = "A" Then
            GCSEgradeweighting = GCSEGradeweighting(s, dt2)
            predictedvalue = 52 * GCSEgradeweighting
            count = count + 1
        End If
        If dt2.Rows(s)("Grade").ToString() = "B" Then
            GCSEgradeweighting = GCSEGradeweighting(s, dt2)
            predictedvalue = 46 * GCSEgradeweighting
            count = count + 1
        End If
        If dt2.Rows(s)("Grade").ToString() = "C" Then
            GCSEgradeweighting = GCSEGradeweighting(s, dt2)
            predictedvalue = 40 * GCSEgradeweighting
            count = count + 1
        End If
        If dt2.Rows(s)("Grade").ToString() = "D" Then
            GCSEgradeweighting = GCSEGradeweighting(s, dt2)
            predictedvalue = 34 * GCSEgradeweighting
            count = count + 1
        End If
    End If
End If

```

```

    End If
    If dt2.Rows(s)("Grade").ToString() = "E" Then
        GCSEgradeweighting = GCSEGradeweighting(s, dt2)
        predictedvalue = 28 * GCSEgradeweighting
        count = count + 1
    End If
    If dt2.Rows(s)("Grade").ToString() = "F" Then
        GCSEgradeweighting = GCSEGradeweighting(s, dt2)
        predictedvalue = 22 * GCSEgradeweighting
        count = count + 1
    End If
    If dt2.Rows(s)("Grade").ToString() = "G" Then
        GCSEgradeweighting = GCSEGradeweighting(s, dt2)
        predictedvalue = 16 * GCSEgradeweighting
        count = count + 1
    End If
    If dt2.Rows(s)("Grade").ToString() = "4" Then
        GCSEgradeweighting = GCSEGradeweighting(s, dt2)
        predictedvalue = 0
    End If
    If dt2.Rows(s)("Grade").ToString() = "5" Then
        GCSEgradeweighting = GCSEGradeweighting(s, dt2)
        predictedvalue = 0
    End If
    'the current predicted total equals the previous currentpredicted total +
    the predicted value
    currentpredictedtotal = currentpredictedtotal + predictedvalue
    Next

    'a mean average point score for the student is created by taking the total of
    all the predicted values and dividing by the number of grades
    averagepointscore = currentpredictedtotal / count

    'the subject variable is set as the value of the subject field
    subject = SubjectField.Text
    'the actual predicted grade is set by passing the subject, row index,
    datatable and average point score to the A2predictedgrade sub in the public module
    predictedgrade = A2predictedgrade(subject, 0, dt, averagepointscore)
    'the predicted grade is then written on the row index of i underneath the
    grade header
    dt3.Rows.Add()
    dt3.Rows(i)("Grade") = predictedgrade
    Next

    'this for next statement loops itself to the last row containing a grade
    'it counts the number of each grade within the dataset
    For i = 0 To (dt3.Rows.Count - 1)
        'if there is a 1 under any of the columns in the dataset for each row then the
        count of that corresponding grade increases by 1

        If dt3.Rows(i)("Grade").ToString() = "A*" Then
            Astargrade = Astargrade + 1
        End If
        If dt3.Rows(i)("Grade").ToString() = "A" Then
            Agrade = Agrade + 1
        End If
        If dt3.Rows(i)("Grade").ToString() = "B" Then
            Bgrade = Bgrade + 1
        End If
        If dt3.Rows(i)("Grade").ToString() = "C" Then
            Cgrade = Cgrade + 1
        End If

```

```

    If dt3.Rows(i)("Grade").ToString() = "D" Then
        Dgrade = Dgrade + 1
    End If
    If dt3.Rows(i)("Grade").ToString() = "E" Then
        Egrade = Egrade + 1
    End If
    If dt3.Rows(i)("Grade").ToString() = "F" Then
        Fgrade = Fgrade + 1
    End If
Next

'sets the value of the text labels within the table that corresponds to the index
value of p
If p = 0 Then
    'the text values of the labels in the table when the tableviewbutton is
    clicked are set to their corresponding number of grades counted from the dataset
    tbl1g.Visible = False
    tbl1ggrade.Visible = False
    Utable1.Text = "U"
    tbl1astar.Text = Astargrade & " out of " & dt3.Rows.Count
    tbl1a.Text = Agrade & " out of " & dt3.Rows.Count
    tbl1b.Text = Bgrade & " out of " & dt3.Rows.Count
    tbl1c.Text = Cgrade & " out of " & dt3.Rows.Count
    tbl1d.Text = Dgrade & " out of " & dt3.Rows.Count
    tbl1e.Text = Egrade & " out of " & dt3.Rows.Count
    tbl1f.Text = Fgrade & " out of " & dt3.Rows.Count
    'this sets the label of the table, so the user knows what the data in the
    table correlates to
    Table1label.Text = "Subject " & p + 1 & ":" & SubjectField.Text & " A2
Predicts " & Yearfield.Text
    'this sets the atoc label text of the table by adding up all the grades from
    A* to c and then dividing by the total number of grades in the dataset, which is then
    converted to a percentage and displayed as a percentage
    tbl1atoc.Text = (Agrade + Bgrade + Cgrade + Astargrade) & " out of " &
dt3.Rows.Count & " (~" & Int(((Agrade + Bgrade + Cgrade + Astargrade) / dt3.Rows.Count) *
100) & "%)"
End If
If p = 1 Then
    tbl2g.Visible = False
    tbl2ggrade.Visible = False
    Utable2.Text = "U"
    tbl2astar.Text = Astargrade & " out of " & dt3.Rows.Count
    tbl2a.Text = Agrade & " out of " & dt3.Rows.Count
    tbl2b.Text = Bgrade & " out of " & dt3.Rows.Count
    tbl2c.Text = Cgrade & " out of " & dt3.Rows.Count
   tbl2d.Text = Dgrade & " out of " & dt3.Rows.Count
   tbl2e.Text = Egrade & " out of " & dt3.Rows.Count
   tbl2f.Text = Fgrade & " out of " & dt3.Rows.Count
    Table2label.Text = "Subject " & p + 1 & ":" & SubjectField.Text & " A2
Predicts " & Yearfield.Text
    tbl2atoc.Text = (Agrade + Bgrade + Cgrade + Astargrade) & " out of " &
dt3.Rows.Count & " (~" & Int(((Agrade + Bgrade + Cgrade + Astargrade) / dt3.Rows.Count) *
100) & "%)"
End If
If p = 2 Then
   tbl3g.Visible = False
   tbl3ggrade.Visible = False
    Utable3.Text = "U"
   tbl3astar.Text = Astargrade & " out of " & dt3.Rows.Count
   tbl3a.Text = Agrade & " out of " & dt3.Rows.Count
   tbl3b.Text = Bgrade & " out of " & dt3.Rows.Count
   tbl3c.Text = Cgrade & " out of " & dt3.Rows.Count

```

```

tbl3d.Text = Dgrade & " out of " & dt3.Rows.Count
tbl3e.Text = Egrade & " out of " & dt3.Rows.Count
tbl3f.Text = Fgrade & " out of " & dt3.Rows.Count
Table3label.Text = "Subject " & p + 1 & ":" & SubjectField.Text & " A2
Predicts " & Yearfield.Text
tbl3atoc.Text = (Agrade + Bgrade + Cgrade + Astargrade) & " out of " &
dt3.Rows.Count & " (~" & Int((Agrade + Bgrade + Cgrade + Astargrade) / dt3.Rows.Count) * 100) & "%"
End If
If p = 3 Then
    tbl4g.Visible = False
    tbl4ggrade.Visible = False
    Utable4.Text = "U"
    tbl4astar.Text = Astargrade & " out of " & dt3.Rows.Count
    tbl4a.Text = Agrade & " out of " & dt3.Rows.Count
    tbl4b.Text = Bgrade & " out of " & dt3.Rows.Count
    tbl4c.Text = Cgrade & " out of " & dt3.Rows.Count
    tbl4d.Text = Dgrade & " out of " & dt3.Rows.Count
    tbl4e.Text = Egrade & " out of " & dt3.Rows.Count
    tbl4f.Text = Fgrade & " out of " & dt3.Rows.Count
    Table4label.Text = "Subject " & p + 1 & ":" & SubjectField.Text & " A2
Predicts " & Yearfield.Text
tbl4atoc.Text = (Agrade + Bgrade + Cgrade + Astargrade) & " out of " &
dt3.Rows.Count & " (~" & Int((Agrade + Bgrade + Cgrade + Astargrade) / dt3.Rows.Count) * 100) & "%"
End If

'this if statement selects what type of graph to display based on the user's selection
If Graphtypefield.Text = "Cumulative Grade Percentage Graph" Then
    'name of the series is set dynamically depending on users choices
    series = "Subject " & p + 1 & ":" & SubjectField.Text & " A2 Predicted " & Yearfield.Text
    'adds a series to the graph
    Subjectpercentageschart.Series.Add(series)
    'adds datapoints to the series of the graph, these points work cumulatively so that each following data point adds all the previous data points values to it
    Subjectpercentageschart.Series(series).Points.AddY(Astargrade / dt3.Rows.Count)
    Subjectpercentageschart.Series(series).Points.AddY((Agrade + Astargrade) / dt3.Rows.Count)
    Subjectpercentageschart.Series(series).Points.AddY((Bgrade + Astargrade + Agrade) / dt3.Rows.Count)
    Subjectpercentageschart.Series(series).Points.AddY((Cgrade + Astargrade + Agrade + Bgrade) / dt3.Rows.Count)
    Subjectpercentageschart.Series(series).Points.AddY((Dgrade + Astargrade + Agrade + Bgrade + Cgrade) / dt3.Rows.Count)
    Subjectpercentageschart.Series(series).Points.AddY((Egrade + Dgrade + Astargrade + Agrade + Bgrade + Cgrade) / dt3.Rows.Count)
    Subjectpercentageschart.Series(series).Points.AddY((Fgrade + Egrade + Dgrade + Astargrade + Agrade + Bgrade + Cgrade) / dt3.Rows.Count)
Else
    'name of the series is set dynamically depending on users choices
    series = "Subject " & p + 1 & ":" & SubjectField.Text & " A2 Predicted " & Yearfield.Text
    'adds a series to the graph
    Subjectpercentageschart.Series.Add(series)
    'adds datapoints to the series of the graph equivalent to the number of grades counted
    Subjectpercentageschart.Series(series).Points.AddY(Astargrade / dt3.Rows.Count)
    Subjectpercentageschart.Series(series).Points.AddY(Agrade / dt3.Rows.Count)

```

```

Subjectpercentageschart.Series(series).Points.AddY(Bgrade / dt3.Rows.Count)
Subjectpercentageschart.Series(series).Points.AddY(Cgrade / dt3.Rows.Count)
Subjectpercentageschart.Series(series).Points.AddY(Dgrade / dt3.Rows.Count)
Subjectpercentageschart.Series(series).Points.AddY(Egrade / dt3.Rows.Count)
Subjectpercentageschart.Series(series).Points.AddY(Fgrade / dt3.Rows.Count)
End If

'sets the labels of the datapoints on the series
Subjectpercentageschart.Series(series).Points(0).AxisLabel = "A*"
Subjectpercentageschart.Series(series).Points(1).AxisLabel = "A"
Subjectpercentageschart.Series(series).Points(2).AxisLabel = "B"
Subjectpercentageschart.Series(series).Points(3).AxisLabel = "C"
Subjectpercentageschart.Series(series).Points(4).AxisLabel = "D"
Subjectpercentageschart.Series(series).Points(5).AxisLabel = "E"
Subjectpercentageschart.Series(series).Points(6).AxisLabel = "U"
End Sub
'this sub runs when the view percentages button is clicked by the user
Private Sub Addsearch_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Addsearch.Click
  'if the state of the txtvalidate function is true then the following code runs
  If textvalidate() = True Then
    'this if statement validates the users selection, since only english language
    or maths are stored for KS2, the program prevents the user from displaying an empty graph
    for KS2 if any other subject is chosen. An appropriate error message is also displayed
    If Levelfield.Text = "KS2" And SubjectField.Text <> "English Language" And
SubjectField.Text <> "Maths" Then
      MsgBox("That subject is not a part of that KeyStage(Level)",
MsgBoxStyle.Information, MessageBoxButtons.OK)
    Else
      'prevents the user from adding more than 4 series
      If p = 4 Then
        MsgBox("You cannot compare any more than 4 Chart series")
      Else
        'the count of the grade variables are reset to 0
        Astargrade = 0
        Agrade = 0
        Bgrade = 0
        Cgrade = 0
        Dgrade = 0
        Egrade = 0
        Fgrade = 0
        Ggrade = 0
        threegrade = 0
        fourgrade = 0
        fivegrade = 0

        'stores a dataset
        Dim ds As New DataSet
        'stores a datatable
        Dim dt As New DataTable

        'the current graph series and title labels are cleared if they have
already been set
        Subjectpercentageschart.Titles.Clear()
        'the toggle labels button becomes visible
        labelstoggle.Visible = True

        'calls the Loadgraphic class
        Dim objPlsWait As New clsOPAWaitScreen
        objPlsWait.ShowWaitScreen("")
      End If
    End If
  End Sub

```

```

If Gradetypefield.Text = "A2 Predicted" Then
    tablettitle1.Text = "Number of Student's Predicted Grade"
    tablettitle2.Text = "Number of Student's Predicted Grade"
    tablettitle3.Text = "Number of Student's Predicted Grade"
    tablettitle4.Text = "Number of Student's Predicted Grade"

    'sets the chart label text depending on the users graphtype
selection
    If Graphtypefield.Text = "Cumulative Grade Percentage Graph" Then
        chartlabel.Text = "Cumulative Predictions by Grade (%)"
    Else
        chartlabel.Text = "Percentage of Students Predicted a Grade
(%)"
    End If

    'selects all students within the year the user has selected in the
yearfield
    Dim sql2 As String
    sql2 = "SELECT DISTINCT Student.StudentID, Student.Firstname,
Student.Surname FROM StudentGroup INNER JOIN Class ON StudentGroup.ClassID = Class.ClassID
INNER JOIN Grade INNER JOIN[Level] ON Grade.LevelID = [Level].LevelID INNER JOIN GradeType
ON Grade.GradeTypeID = GradeType.GradeTypeID INNER JOIN Student ON Grade.StudentID =
Student.StudentID ON StudentGroup.StudentID = Student.StudentID INNER JOIN Subject ON
Grade.SubjectID = Subject.SubjectID INNER JOIN DatasetID ON Grade.DatasetID =
DatasetID.DatasetID WHERE Student.DatasetID = '" & Yearfield.SelectedValue & "' ORDER BY
Student.StudentID ASC"
    'passes the SQL statement to the A2predictions sub
    A2Predictions(sql2)

Else
    'sets the chart label text depending on the users graphtype
selection
    If Graphtypefield.Text = "Cumulative Grade Percentage Graph" Then
        chartlabel.Text = "Cumulative Percentage by Grade (%)"
    Else
        chartlabel.Text = "Percentage of Students Achieving a Grade
(%)"
    End If

    If Levelfield.Text = "KS4" Then
        'if the levelfield value is KS4 then this code runs
        'runs the KS4 percentages sub
        KS4percentages()
        'the graph becomes visible
        Subjectpercentageschart.Visible = True
    Else
        'runs the KS2 percentages sub
        KS2percentages()
        'the graph becomes visible
        Subjectpercentageschart.Visible = True
    End If
End If
'disables the legend from the graph
Subjectpercentageschart.Series(p).IsVisibleInLegend = True
'shows the values of datapoints
Subjectpercentageschart.Series(p).IsValueShownAsLabel = True
'sets the colour of the datapoint labels as maroon
Subjectpercentageschart.Series(p).ChartType =
DataVisualization.Charting.SeriesChartType.Line
'changes the chart colour to skyblue

```

```

Subjectpercentageschart.Series(p).LabelForeColor = Color.Maroon
'changes the chart to a 3D bar style
Subjectpercentageschart.Series(0).Color = Color.DarkGreen
'sets the minimum and maximum axis
Subjectpercentageschart.ChartAreas(0).AxisY.Minimum = 0
Subjectpercentageschart.ChartAreas(0).AxisY.Maximum = 1
'sets the format of the chart labels
Subjectpercentageschart.ChartAreas(0).AxisY.LabelStyle.Format = "#%"
'sets the format of the series labels
Subjectpercentageschart.Series(p).LabelFormat = "##.##" & "%"
'sets the colour of the axis labels
Subjectpercentageschart.ChartAreas(0).AxisX.LabelStyle.ForeColor =
Color.Maroon
Color.Maroon
Subjectpercentageschart.ChartAreas(0).AxisY.LabelStyle.ForeColor =
Color.Maroon
'sets the background of the chart as transparent
Subjectpercentageschart.ChartAreas(0).BackColor = Color.Transparent
'increments the value of p by 1 as a new series is added by the user
p = p + 1
'graph becomes visible
Subjectpercentageschart.Visible = True
'if the user selects KS4 then the groupview groupbox is visible
If Levelfield.Text = "KS4" Then
    Groupview.Visible = True
End If

'these fields are no longer enables so that the user cannot add a series
for KS2 and KS4, or add a series of one graph type and a series for the other graph type
Gradetypefield.Enabled = False
Graphtypefield.Enabled = False
Levelfield.Enabled = False

'closes the loadgraphic wait and resets its objplswait variable
objPlsWait.CloseWaitScreen()
objPlsWait = Nothing
Addsearch.Text = "Add Series"
End If
End If
Else
    'if the state of the txtvalidate function is not true then the following error
message displays and nothing else happens
    MsgBox("One or more textboxes have not been completed",
MsgBoxStyle.Information, MessageBoxButtons.OK)
End If

End Sub
'this sub is run when the Clear Selection button is clicked
Private Sub Clearselection_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Clearselection.Click
    'clears all controls on this form
    Me.Controls.Clear()
    're-initializes all components of this form
    InitializeComponent()
    'then re-loads this form with all components reset
    CompareSubject_Percentages(e, e)
End Sub
'this sub runs when the toggle labels button is clicked by the user
Private Sub labelstoggle_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles labelstoggle.Click
    'stores an index variable
    Dim s As Integer
    'if the user wishes to turn chart labels off then the following code runs

```

```

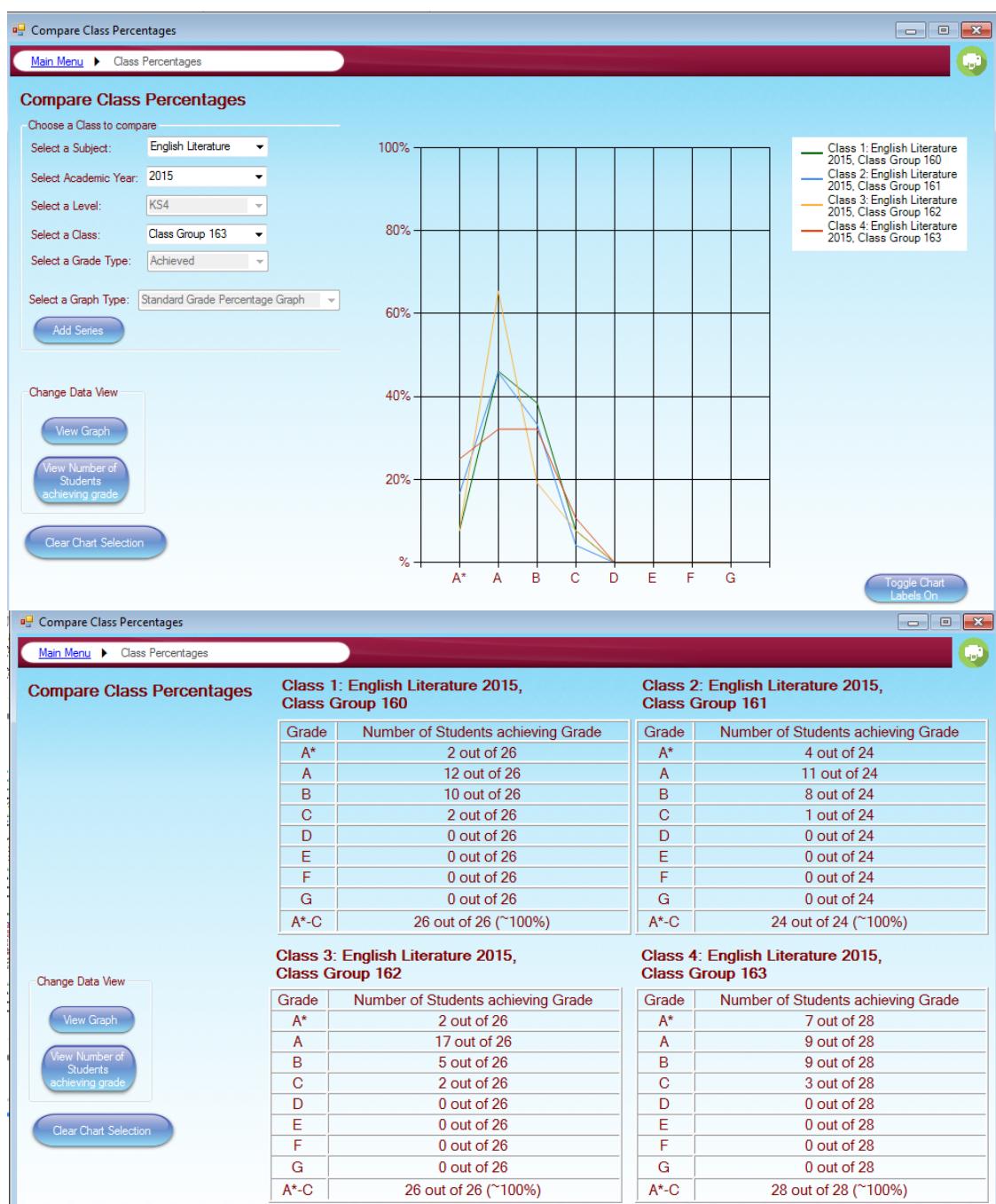
If labelstoggle.Text = "Toggle Chart Labels Off" Then
    'this for next statemnt goes through every series that has been added by the
    user and turns off labels for that series
    For s = 0 To (p - 1)
        Subjectpercentageschart.Series(s).IsValueShownAsLabel = False
    Next
    'changes the text of the button
    labelstoggle.Text = "Toggle Chart Labels On"
    'but if the user wishes to turn chart labels on then the following code runs
ElseIf labelstoggle.Text = "Toggle Chart Labels On" Then
    For s = 0 To (p - 1)
        Subjectpercentageschart.Series(s).IsValueShownAsLabel = True
    Next
    'changes the text of the button
    labelstoggle.Text = "Toggle Chart Labels Off"
End If
End Sub
'this sub runs when the view graph button is clicked by the user
Private Sub Graphview_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Graphview.Click
    'the following objects are set to either visible or non-visible
    'all tables and table headings are no longer visible
    Subjectpercentageschart.Visible = True
    Comparesubjectgroup.Visible = True
    labelstoggle.Visible = True
    Table1label.Visible = False
    Table2label.Visible = False
    Table3label.Visible = False
    Table4label.Visible = False
    Seriestable1.Visible = False
    Seriestable2.Visible = False
    Seriestable3.Visible = False
    Seriestable4.Visible = False
End Sub
'this sub runs when the view number of students... button is clicked by the user
Private Sub TablesView_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles TablesView.Click
    'the following objects are set to non-visible
    Subjectpercentageschart.Visible = False
    Comparesubjectgroup.Visible = False
    labelstoggle.Visible = False
    'the following tables and table headings become visible depending on the value of
    p as it is incremented
    Select Case (p)
        Case 1
            Table1label.Visible = True
            Seriestable1.Visible = True
        Case 2
            Table1label.Visible = True
            Seriestable1.Visible = True
            Table2label.Visible = True
            Seriestable2.Visible = True
        Case 3
            Table1label.Visible = True
            Seriestable1.Visible = True
            Table2label.Visible = True
            Seriestable2.Visible = True
            Seriestable3.Visible = True
            Table3label.Visible = True
        Case 4
            Table1label.Visible = True
            Seriestable1.Visible = True
    End Select
End Sub

```

```
Table2label.Visible = True
Seriestable2.Visible = True
Seriestable3.Visible = True
Table3label.Visible = True
Table4label.Visible = True
Seriestable4.Visible = True
End Select
End Sub
' deals with the print form button
Private Sub printbutton_Click(sender As System.Object, e As System.EventArgs) Handles printbutton.Click
    'sets the printer settings to default
    PrintForm.PrinterSettings = PrintDialog.PrinterSettings
    If PrintDialog.ShowDialog() = DialogResult.OK Then
        'allows the user to change print settings
        PrintForm.PrinterSettings = PrintDialog.PrinterSettings
        'changes the orientation of the printed document to landscape
        Me.PrintForm.PrinterSettings.DefaultPageSettings.Landscape = True
        'refreshes the form so that the printdialog is not printed along with the form
        Me.Refresh()
        'prints the current form
        Me.PrintForm.Print()
        'Percentageschart.Printing.Print(True)
    End If
End Sub
'this sub determines whether the gradetypefield is enabled, depending on hte user's
level choice.
Private Sub Levelefield_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Levelefield.SelectedIndexChanged
    If Levelefield.Text = "KS2" Then
        Gradetypefield.Enabled = False
        Gradetypefield.Text = "Achieved"
    Else
        Gradetypefield.Enabled = True
    End If
End Sub
End Class
```

Compare Class Percentages

This form allows a user to compare grade statistics of achieved GCSE grades or predicted A2 grades. This is done by comparing individual class performance within years, a user can select which class they want to compare and which grade type and level they want to compare. For example a user could compare four different maths class groups in the same year. A maximum of four subjects can be compared at any one time. The statistics can be displayed as percentages on a linear line graph or a cumulative grade line graph. There is also a viewing mode that allows the user to view the raw statistics of students achieving or predicted individual grades and the number of students achieving or predicted A* - C grades. At any time the current classes that are being compared can be cleared so that a new set of classes can be compared. A printed copy of this form can be produced by using the print icon, which is very useful as a copy of the raw statistics can be produced or a copy of the graph comparisons can be produced.



Addsearch	System.Windows.Forms.Button
banner	System.Windows.Forms.PictureBox
chartlabel	System.Windows.Forms.Label
Classespcentageschart	System.Windows.Forms.DataVisualization.Charting.Chart
Classgroupfield	System.Windows.Forms.ComboBox
Classlabel	System.Windows.Forms.Label
classpercentageslink	System.Windows.Forms.LinkLabel
Clearselection	System.Windows.Forms.Button
Compare_Class_Percentages	System.Windows.Forms.Form
Comparesubjectgroup	System.Windows.Forms.GroupBox
Gradetypefield	System.Windows.Forms.ComboBox
Graphtypefield	System.Windows.Forms.ComboBox
Graphview	System.Windows.Forms.Button
Groupview	System.Windows.Forms.GroupBox
labelstoggle	System.Windows.Forms.Button
Levelfield	System.Windows.Forms.ComboBox
Levelelabel	System.Windows.Forms.Label
MainMenu	System.Windows.Forms.LinkLabel
printbutton	System.Windows.Forms.Button
PrintDialog	System.Windows.Forms.PrintDialog
PrintForm	Microsoft.VisualBasic.PowerPacks.Printing.PrintForm
Seriestable1	System.Windows.Forms.TableLayoutPanel
Seriestable2	System.Windows.Forms.TableLayoutPanel
Seriestable3	System.Windows.Forms.TableLayoutPanel
Seriestable4	System.Windows.Forms.TableLayoutPanel
SubjectField	System.Windows.Forms.ComboBox
subjectlabel	System.Windows.Forms.Label
Table1label	System.Windows.Forms.Label
Table2label	System.Windows.Forms.Label
Table3label	System.Windows.Forms.Label
Table4label	System.Windows.Forms.Label
TablesView	System.Windows.Forms.Button
tabletitle1	System.Windows.Forms.Label
tabletitle2	System.Windows.Forms.Label
tabletitle3	System.Windows.Forms.Label
tabletitle4	System.Windows.Forms.Label
tbl1a	System.Windows.Forms.Label
tbl1astar	System.Windows.Forms.Label
tbl1atoc	System.Windows.Forms.Label
tbl1b	System.Windows.Forms.Label
tbl1c	System.Windows.Forms.Label
tbl1d	System.Windows.Forms.Label
tbl1e	System.Windows.Forms.Label
tbl1f	System.Windows.Forms.Label
tbl2a	System.Windows.Forms.Label
tbl2astar	System.Windows.Forms.Label
tbl2atoc	System.Windows.Forms.Label
tbl2b	System.Windows.Forms.Label
tbl2c	System.Windows.Forms.Label
tbl2d	System.Windows.Forms.Label
tbl2e	System.Windows.Forms.Label
tbl2f	System.Windows.Forms.Label
tbl3a	System.Windows.Forms.Label
tbl3astar	System.Windows.Forms.Label
tbl3atoc	System.Windows.Forms.Label
tbl3b	System.Windows.Forms.Label
tbl3c	System.Windows.Forms.Label
tbl3d	System.Windows.Forms.Label
tbl3e	System.Windows.Forms.Label
tbl3f	System.Windows.Forms.Label
tbl4a	System.Windows.Forms.Label
tbl4astar	System.Windows.Forms.Label
tbl4atoc	System.Windows.Forms.Label
tbl4b	System.Windows.Forms.Label
tbl4c	System.Windows.Forms.Label
tbl4d	System.Windows.Forms.Label
tbl4e	System.Windows.Forms.Label
tbl4f	System.Windows.Forms.Label
Yearfield	System.Windows.Forms.ComboBox
yearlabel	System.Windows.Forms.Label

Parameter Name	Description
CompareClass_Percentages	This sub connects to the database server when the form initially loads and dynamically generates the list properties of all the combo boxes on the form and sets their value members. It also sets the default text property of the graph type combo box.
MainMenu_LinkClicked	When the main menu link label is clicked all open forms close and the main menu displays.
textvalidate	This function sets a true or false condition depending on whether any textboxes on the form are blank. It also sets the background colour of textboxes appropriately to indicate to the user that fields have not been completed.

ks4percentages	Selects all grades within the database and then counts the number of each KS4 grade e.g. A's B's. It then sets the text properties of the values within the statistics table and adds data points to the series of the chart, these data points depend on the user's selection of a standard or cumulative graph type. It then adds the corresponding data point axis labels for each KS4 grade.
KS2percentages	Selects all grades within the database and then counts the number of each KS2 grade e.g. 3's 4's. It then sets the text properties of the values within the statistics table and adds data points to the series of the chart, these data points depend on the user's selection of a standard or cumulative graph type. It then adds the corresponding data point axis labels for each KS2 grade.
A2Predictions	This sub generates a series of predicted grades by first selecting all students that match the filter criteria, and calculating an average point score for each student based on their GCSE grades. From this average point score a predicted grade is then generated for the subject the user wants to view predicted class grades for. Once all of the predicted grades have been generated, data points are added to the chart series, these are either linear data points or cumulative data points depending on the users graph choice. The label properties of the statistics table are also set as the number of each predicted grade. Corresponding data point axis labels for each predicted grade are also added to the chart.
Addsearch_Click	When the add series button is clicked by the user, this sub determines which type of series to add to the chart. It is also responsible for validation and formatting the chart.
Clearselection_Click	Creates a new instance of every object on the form (essentially re-loads the form).
labelstoggle_Click	Handles the button click event of the Toggle chart labels button. When clicked an index variable loops through all series and changes the visible property of the labels to false. It also changes the text property of the button.
Graphview_Click	When the view graph button is clicked all objects associated with the graph visible to the user and all objects associated with the statistics tables non-visible to the user.
TablesView_Click	When the view table's button is clicked all objects associated with the graph become non-visible to the user and all objects associated with

	the statistics tables become visible to the user.
printbutton_Click	This sub allows the user to print a hard copy of the form when the print icon is clicked, additional print dialog options are then displayed allowing the user to choose printer and number of copies amongst other settings.
Levelfield_SelectedIndexChanged	Whenever the text value of the level field is changed this sub determines whether the grade type field is either enabled or not enabled.
Classfieldgenerate	This sub queries the database using the user's selected class filters to return all the classes that match these filters. The returned results are then set as the list property of the classgroupfield combo box. A user can then select one of these classes as their choice of class to compare.
Yearfield_SelectedIndexChanged	This sub handles every instance the year field is altered by the user's selection. It validates whether all of the required text fields have been completed and if so runs the classfieldgenerate sub.
SubjectField_SelectedIndexChanged	This sub handles every instance the subject field is altered by the user's selection. It validates whether all of the required text fields have been completed and if so runs the classfieldgenerate sub.

```

'Imports the following system code commands so that I can use them appropriately
Imports System.Data.SqlClient
Imports System.Data
Imports System.Configuration
Imports System.Data.DataTable

Public Class Compare_Class_Percentages
    'stores series as an empty string that can be used within this class
    Dim series As String = Nothing
    'i and p are used as an index variable in my for next statements
    Dim i As Integer = 0
    'p stores teh index of the series to add to the chart
    Dim p As Integer = 0
    'these variables store the number of grades counted
    Dim threegrade As Integer = 0
    Dim fourgrade As Integer = 0
    Dim fivegrade As Integer = 0
    Dim Astargrade As Integer
    Dim Agrade As Integer = 0
    Dim Bgrade As Integer = 0
    Dim Cgrade As Integer = 0
    Dim Dgrade As Integer = 0
    Dim Egrade As Integer = 0
    Dim Fgrade As Integer = 0
    Dim Ggrade As Integer = 0
    'this sub runs when the form initally loads
    Private Sub CompareClass_Percentages(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Me.Load
        'runs the databaseconnect sub from the public module
        DatabaseConnect()
        'stores dt as datatable
        Dim dt As DataTable

        p = 0

        'Uses the Generate combo function of the module to populate a combobox by passing
        the table, field and needempty variables to the Generatecombo function and then returning
        the result in a datatable
        dt = GenerateCombo("Subject", "SubjectID, Subject", True)
        'sets the datasource of the combobox
        SubjectField.DataSource = dt
        'sets the display member of the combobox
        SubjectField.DisplayMember = "Subject"
        'the index of each item in the combobox is set as their corresponding ID from the
        database
        SubjectField.ValueMember = "SubjectID"

        dt = GenerateCombo("DatasetID", "DatasetID, Year", True)
        Yearfield.DataSource = dt
        Yearfield.DisplayMember = "Year"
        Yearfield.ValueMember = "DatasetID"

        dt = GenerateCombo("Level", "levelID, Level", True)
        Levelfield.DataSource = dt
        Levelfield.DisplayMember = "Level"
        Levelfield.ValueMember = "LevelID"

        dt = GenerateCombo("GradeType", "GradeTypeID, GradeType", True)
        Gradetypefield.DataSource = dt
        Gradetypefield.DisplayMember = "GradeType"
        Gradetypefield.ValueMember = "GradeTypeID"

```

```

'sets the default text of the grapgtypefield
Graphtypefield.Text = "Standard Grade Percentage Graph"
End Sub
'this sub runs when the main menu link is clicked
Private Sub MainMenu_LinkClicked(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles MainMenu.LinkClicked
    'the following forms either close or open
    Main_Menu.Show()
    Me.Close()
End Sub

'this function returns a true or false state depending on whether the conditions
within this sub have been met or not
Public Function textvalidate() As Boolean
    'initially the functions state is set to true
    textvalidate = True
    'if any of the textboxes are left blank then the function returns a false state
    and the background of the textbox field changes to red, otherwise the background of the
    textbox changes to white
    If Classgroupfield.Text = "" Then
        textvalidate = False
        Classgroupfield.BackColor = Color.Salmon
    Else
        Classgroupfield.BackColor = Color.White
    End If
    If Gradetypefield.Text = "" Then
        textvalidate = False
        Gradetypefield.BackColor = Color.Salmon
    Else
        Gradetypefield.BackColor = Color.White
    End If
    If SubjectField.Text = "" Then
        textvalidate = False
        SubjectField.BackColor = Color.Salmon
    Else
        SubjectField.BackColor = Color.White
    End If
    If Yearfield.Text = "" Then
        textvalidate = False
        Yearfield.BackColor = Color.Salmon
    Else
        Yearfield.BackColor = Color.White
    End If
    If Levelfield.Text = "" Then
        textvalidate = False
        Levelfield.BackColor = Color.Salmon
    Else
        Levelfield.BackColor = Color.White
    End If
End Function
'this sub runs when all of the fields that require the user's input have been
completed
Private Sub Classfieldgenerate()
    'selects the classes based on the user's choice of subject and year
    sql = "SELECT Distinct Class.ClassID, Class.ClassGroup FROM Department INNER JOIN
Subject ON Department.DepartmentID = Subject.DepartmentID INNER JOIN DatasetID INNER JOIN
Class ON DatasetID.DatasetID = Class.DatasetID ON Subject.SubjectID = Class.SubjectID
INNER JOIN StudentGroup ON Class.ClassID = StudentGroup.ClassID WHERE Class.SubjectID = ''
& SubjectField.SelectedValue & '' And Class.DatasetID = '' & Yearfield.SelectedValue & ''
ORDER BY Class.ClassID ASC"
    'stores a SQL command that uses the SQL statement to query the database that is
found when following the connection string of con

```

```

Using comm As SqlCommand = New SqlCommand(sql, con)
    'stores a SQL dataadareader which can read the contents of the sql query from
the database
    Dim rs As SqlDataReader = comm.ExecuteReader
    'stores a datatable
    Dim dt As DataTable = New DataTable
    'the datatable is filled with the values that the datareader reads
    dt.Load(rs)

        'the index of each item(class) in the combobox is set as its corresoinding ID
in the database
        Classgroupfield.ValueMember = "ClassID"
        'sets the displaymemeber of the combobox
        Classgroupfield.DisplayMember = "ClassGroup"
        'sets the datasource of the combobox
        Classgroupfield.DataSource = dt
End Using
'makes the class combobox and its label visible to the user
Classlabel.Visible = True
Classgroupfield.Visible = True
End Sub
'this sub runs when the text within the level field is altered
'this sub works in a similar manner to the two subs below
Private Sub Levelfield_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Levelfield.SelectedIndexChanged
    'if the field is blank then the class combobox and label are not visible
    If Levelfield.Text = "" Then
        Classlabel.Visible = False
        Classgroupfield.Visible = False
        'but if the followind three text fields are not blank then the
classfieldgenerate sub is called which fills a combobox with the users selection of
classes
    ElseIf Levelfield.Text <> "" And Yearfield.Text <> "" And SubjectField.Text <> ""
Then
        Classfieldgenerate()
    End If
    If Levelfield.Text = "KS2" Then
        Gradetypefield.Enabled = False
        Gradetypefield.Text = "Achieved"
    Else
        Gradetypefield.Enabled = True
    End If
End Sub
'this sub runs when the text within the year field is altered
Private Sub Yearfield_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Yearfield.SelectedIndexChanged
    If Yearfield.Text = "" Then
        Classlabel.Visible = False
        Classgroupfield.Visible = False
    ElseIf Levelfield.Text <> "" And Yearfield.Text <> "" And SubjectField.Text <> ""
Then
        Classfieldgenerate()
    End If
End Sub
'this sub runs when the text within the subject field is altered
Private Sub SubjectField_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles SubjectField.SelectedIndexChanged
    If SubjectField.Text = "" Then
        Classlabel.Visible = False
        Classgroupfield.Visible = False
    ElseIf Levelfield.Text <> "" And Yearfield.Text <> "" And SubjectField.Text <> ""
Then

```

```

        Classfieldgenerate()
    End If
End Sub
'this sub is run when the Clear Selection button is clicked
Private Sub Clearselection_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Clearselection.Click
    'clears all controls on this form
    Me.Controls.Clear()
    're-initializes all components of this form
    InitializeComponent()
    'then re-loads this form with all components reset
    CompareClass_Percentages(e, e)
End Sub
'this sub runs when the toggle labels button is clicked by the user
Private Sub labelstoggle_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles labelstoggle.Click
    'stores an index variable
    Dim s As Integer

    'if the user wishes to turn chart labels off then the following code runs
    If labelstoggle.Text = "Toggle Chart Labels Off" Then
        'this for next statemnt goes through every series that has been added by the
        user and turns off labels for that series
        For s = 0 To (p - 1)
            Classespercentageschart.Series(s).IsValueShownAsLabel = False
        Next
        'changes the text of the button
        labelstoggle.Text = "Toggle Chart Labels On"

        'but if the user wishes to turn chart labels on then the following code runs
    ElseIf labelstoggle.Text = "Toggle Chart Labels On" Then
        For s = 0 To (p - 1)
            Classespercentageschart.Series(s).IsValueShownAsLabel = True
        Next
        'changes the text of the button
        labelstoggle.Text = "Toggle Chart Labels Off"
    End If

End Sub
'this sub runs when the view number of students... button is clicked by the user
Private Sub TablesView_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles TablesView.Click
    'the following objects are set to non-visible
    Classespercentageschart.Visible = False
    Comparesubjectgroup.Visible = False
    labelstoggle.Visible = False

    'the following tables and table headings become visible depending on the value of
    p as it is incremented
    Select Case (p)
        Case 1
            Table1label.Visible = True
            Seriestable1.Visible = True
        Case 2
            Table1label.Visible = True
            Seriestable1.Visible = True
            Table2label.Visible = True
            Seriestable2.Visible = True
        Case 3
            Table1label.Visible = True
            Seriestable1.Visible = True
            Table2label.Visible = True
    End Select
End Sub

```

```

        Seriestable2.Visible = True
        Seriestable3.Visible = True
        Table3label.Visible = True
    Case 4
        Table1label.Visible = True
        Seriestable1.Visible = True
        Table2label.Visible = True
        Seriestable2.Visible = True
        Seriestable3.Visible = True
        Table3label.Visible = True
        Table4label.Visible = True
        Seriestable4.Visible = True
    End Select
End Sub
'this sub runs when the view graph button is clicked by the user
Private Sub Graphview_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Graphview.Click
    'the following objects are set to either visible or non-visible
    'all tables and table headings are no longer visible
    Classespercentageschart.Visible = True
    Comparesubjectgroup.Visible = True
    labelstoggle.Visible = True
    Table1label.Visible = False
    Table2label.Visible = False
    Table3label.Visible = False
    Table4label.Visible = False
    Seriestable1.Visible = False
    Seriestable2.Visible = False
    Seriestable3.Visible = False
    Seriestable4.Visible = False
End Sub
'this sub deals with KS4 percentages
Public Sub kS4percentages()
    'stores ds as a dataset
    Dim ds As New DataSet
    'stores dt as a datatable
    Dim dt As New DataTable

    ' This SQL statement fills a dataset depending on the users class choice
    sql = "SELECT Grade.Grade, Grade.DatasetID, Grade.LevelID, Class.ClassGroup FROM
StudentGroup INNER JOIN Subject INNER JOIN Class ON Subject.SubjectID = Class.SubjectID
INNER JOIN Student INNER JOIN Grade ON Student.StudentID = Grade.StudentID ON
Subject.SubjectID = Grade.SubjectID ON StudentGroup.ClassID = Class.ClassID AND
StudentGroup.StudentID = Student.StudentID WHERE Class.ClassGroup = '" &
Classgroupfield.Text & "'"
    'the dataset is set after passing the following SQL statement and table into the
GenerateDataset sub of the public module
    ds = generateDataset(sql, "temp")
    'the datatable contents is set as the returned dataset once the SQL statement has
filled the dataset
    dt = ds.Tables(0)

    'this for next statement loops itself to the last row containing a grade
    'it counts the number of each grade within the dataset
    For i = 0 To (dt.Rows.Count - 1)
        'if there is a matching grade under any of the columns in the dataset for each
        row then the count of that corresponding grade count increases by 1
        If dt.Rows(i)("Grade").ToString() = "A*" Then
            Astargrade = Astargrade + 1
        End If
        If dt.Rows(i)("Grade").ToString() = "A" Then
            Agrade = Agrade + 1
        End If
    Next
End Sub

```

```

        End If
        If dt.Rows(i)("Grade").ToString() = "B" Then
            Bgrade = Bgrade + 1
        End If
        If dt.Rows(i)("Grade").ToString() = "C" Then
            Cgrade = Cgrade + 1
        End If
        If dt.Rows(i)("Grade").ToString() = "D" Then
            Dgrade = Dgrade + 1
        End If
        If dt.Rows(i)("Grade").ToString() = "E" Then
            Egrade = Egrade + 1
        End If
        If dt.Rows(i)("Grade").ToString() = "F" Then
            Fgrade = Fgrade + 1
        End If
        If dt.Rows(i)("Grade").ToString() = "G" Then
            Ggrade = Ggrade + 1
        End If
    Next
    'sets the value of the text labels within the table that corresponds to the index
    value of p
    If p = 0 Then
        'the text values of the labels in the table when the tableviewbutton is
        clicked are set to their corresponding number of grades counted from the dataset
        tbl1astar.Text = Astargrade & " out of " & dt.Rows.Count
        tbl1a.Text = Agrade & " out of " & dt.Rows.Count
        tbl1b.Text = Bgrade & " out of " & dt.Rows.Count
        tbl1c.Text = Cgrade & " out of " & dt.Rows.Count
        tbl1d.Text = Dgrade & " out of " & dt.Rows.Count
        tbl1e.Text = Egrade & " out of " & dt.Rows.Count
        tbl1f.Text = Fgrade & " out of " & dt.Rows.Count
        tbl1ggrade.Text = Ggrade & " out of " & dt.Rows.Count
        'this sets the label of the table, so the user knows what the data in the
        table correlates to
        Table1label.Text = "Class " & p + 1 & ":" & SubjectField.Text & " " &
        Yearfield.Text & ", " & vbCrLf & Classgroupfield.Text
        'this sets the atoc label text of the table by adding up all the grades from
        A* to c and then dividing by the total number of grades in the dataset, which is then
        converted to a percentage and displayed as a percentage
        tbl1atoc.Text = (Agrade + Bgrade + Cgrade + Astargrade) & " out of " &
        dt.Rows.Count & " (~" & Int(((Agrade + Bgrade + Cgrade + Astargrade) / dt.Rows.Count) *
        100) & "%)"
    End If
    If p = 1 Then
        tbl2astar.Text = Astargrade & " out of " & dt.Rows.Count
        tbl2a.Text = Agrade & " out of " & dt.Rows.Count
        tbl2b.Text = Bgrade & " out of " & dt.Rows.Count
        tbl2c.Text = Cgrade & " out of " & dt.Rows.Count
        tbl2d.Text = Dgrade & " out of " & dt.Rows.Count
        tbl2e.Text = Egrade & " out of " & dt.Rows.Count
        tbl2f.Text = Fgrade & " out of " & dt.Rows.Count
        tbl2ggrade.Text = Ggrade & " out of " & dt.Rows.Count
        Table2label.Text = "Class " & p + 1 & ":" & SubjectField.Text & " " &
        Yearfield.Text & ", " & vbCrLf & Classgroupfield.Text
        tbl2atoc.Text = (Agrade + Bgrade + Cgrade + Astargrade) & " out of " &
        dt.Rows.Count & " (~" & Int(((Agrade + Bgrade + Cgrade + Astargrade) / dt.Rows.Count) *
        100) & "%)"
    End If
    If p = 2 Then
        tbl3astar.Text = Astargrade & " out of " & dt.Rows.Count

```

```

tbl3a.Text = Agrade & " out of " & dt.Rows.Count
tbl3b.Text = Bgrade & " out of " & dt.Rows.Count
tbl3c.Text = Cgrade & " out of " & dt.Rows.Count
tbl3d.Text = Dgrade & " out of " & dt.Rows.Count
tbl3e.Text = Egrade & " out of " & dt.Rows.Count
tbl3f.Text = Fgrade & " out of " & dt.Rows.Count
tbl3ggrade.Text = Ggrade & " out of " & dt.Rows.Count
Table3label.Text = "Class " & p + 1 & ":" & SubjectField.Text & " " &
Yearfield.Text & ", " & vbCrLf & Classgroupfield.Text
tbl3atoc.Text = (Agrade + Bgrade + Cgrade + Astargrade) & " out of " &
dt.Rows.Count & " (~" & Int(((Agrade + Bgrade + Cgrade + Astargrade) / dt.Rows.Count) *
100) & "%)"
End If
If p = 3 Then
    tbl4astar.Text = Astargrade & " out of " & dt.Rows.Count
    tbl4a.Text = Agrade & " out of " & dt.Rows.Count
    tbl4b.Text = Bgrade & " out of " & dt.Rows.Count
    tbl4c.Text = Cgrade & " out of " & dt.Rows.Count
    tbl4d.Text = Dgrade & " out of " & dt.Rows.Count
    tbl4e.Text = Egrade & " out of " & dt.Rows.Count
    tbl4f.Text = Fgrade & " out of " & dt.Rows.Count
    tbl4ggrade.Text = Ggrade & " out of " & dt.Rows.Count
    Table4label.Text = "Class " & p + 1 & ":" & SubjectField.Text & " " &
Yearfield.Text & ", " & vbCrLf & Classgroupfield.Text
    tbl4atoc.Text = (Agrade + Bgrade + Cgrade + Astargrade) & " out of " &
dt.Rows.Count & " (~" & Int(((Agrade + Bgrade + Cgrade + Astargrade) / dt.Rows.Count) *
100) & "%)"
End If
'this if statement selects what type of graph to display based on the user's
selection
If Graphtypefield.Text = "Cumulative Grade Percentage Graph" Then
    'name of the series is set dynamically depending on users choices
    series = "Class " & p + 1 & ":" & SubjectField.Text & " " & Yearfield.Text &
", " & Classgroupfield.Text
    'adds a series to the graph
    Classespercentageschart.Series.Add(series)
    'adds datapoints to the series of the graph, these points work cumulatively so
    that each following data point adds all the previous data points values to it
    Classespercentageschart.Series(series).Points.AddY(Astargrade / dt.Rows.Count)
    Classespercentageschart.Series(series).Points.AddY((Agrade + Astargrade) /
    dt.Rows.Count)
    Classespercentageschart.Series(series).Points.AddY((Bgrade + Astargrade +
    Agrade) / dt.Rows.Count)
    Classespercentageschart.Series(series).Points.AddY((Cgrade + Astargrade +
    Agrade + Bgrade) / dt.Rows.Count)
    Classespercentageschart.Series(series).Points.AddY((Dgrade + Astargrade +
    Agrade + Bgrade + Cgrade) / dt.Rows.Count)
    Classespercentageschart.Series(series).Points.AddY((Egrade + Dgrade +
    Astargrade + Agrade + Bgrade + Cgrade) / dt.Rows.Count)
    Classespercentageschart.Series(series).Points.AddY((Fgrade + Egrade + Dgrade +
    Astargrade + Agrade + Bgrade + Cgrade) / dt.Rows.Count)
    Classespercentageschart.Series(series).Points.AddY((Ggrade + Fgrade + Egrade +
    Dgrade + Astargrade + Agrade + Bgrade + Cgrade) / dt.Rows.Count)
Else
    'name of the series is set dynamically depending on users choices
    series = "Class " & p + 1 & ":" & SubjectField.Text & " " & Yearfield.Text &
", " & Classgroupfield.Text
    'adds a series to the graph
    Classespercentageschart.Series.Add(series)
    'adds datapoints to the series of the graph equivalent to the number of grades
    counted
    Classespercentageschart.Series(series).Points.AddY(Astargrade / dt.Rows.Count)

```

```

    Classespercentageschart.Series(series).Points.AddY(Agrade / dt.Rows.Count)
    Classespercentageschart.Series(series).Points.AddY(Bgrade / dt.Rows.Count)
    Classespercentageschart.Series(series).Points.AddY(Cgrade / dt.Rows.Count)
    Classespercentageschart.Series(series).Points.AddY(Dgrade / dt.Rows.Count)
    Classespercentageschart.Series(series).Points.AddY(Egrade / dt.Rows.Count)
    Classespercentageschart.Series(series).Points.AddY(Fgrade / dt.Rows.Count)
    Classespercentageschart.Series(series).Points.AddY(Ggrade / dt.Rows.Count)
End If
'sets the labels of the datapoints on the series
Classespercentageschart.Series(series).Points(0).AxisLabel = "A*"
Classespercentageschart.Series(series).Points(1).AxisLabel = "A"
Classespercentageschart.Series(series).Points(2).AxisLabel = "B"
Classespercentageschart.Series(series).Points(3).AxisLabel = "C"
Classespercentageschart.Series(series).Points(4).AxisLabel = "D"
Classespercentageschart.Series(series).Points(5).AxisLabel = "E"
Classespercentageschart.Series(series).Points(6).AxisLabel = "F"
Classespercentageschart.Series(series).Points(7).AxisLabel = "G"
End Sub
'this sub deals with KS2 percentages and works similarly to the Sub above
Public Sub KS2percentages()
    Dim ds As New DataSet
    Dim dt As New DataTable

        ' This SQL statement fills a dataset depending on the users class choice
        sql = "SELECT Grade.Grade, Grade.DatasetID, Grade.LevelID, Class.ClassGroup FROM
StudentGroup INNER JOIN Subject INNER JOIN Class ON Subject.SubjectID = Class.SubjectID
INNER JOIN Student INNER JOIN Grade ON Student.StudentID = Grade.StudentID ON
Subject.SubjectID = Grade.SubjectID ON StudentGroup.ClassID = Class.ClassID AND
StudentGroup.StudentID = Student.StudentID WHERE Class.ClassGroup = '" &
Classgroupfield.Text & "'"
        ds = generateDataset(sql, "temp")
        dt = ds.Tables(0)

        'counts the number of KS2 grades
        For i = 0 To (dt.Rows.Count - 1)
            If dt.Rows(i)("Grade").ToString() = "3" Then
                threegrade = threegrade + 1
            End If
            If dt.Rows(i)("Grade").ToString() = "4" Then
                fourgrade = fourgrade + 1
            End If
            If dt.Rows(i)("Grade").ToString() = "5" Then
                fivegrade = fivegrade + 1
            End If
        Next

        'sets the type of series to add to the graph based on whether the user wants to
        'see a cumulative or non-cumulative graph
        If Graphtypefield.Text = "Cumulative Grade Percentage Graph" Then
            series = "Class " & p + 1 & ":" & SubjectField.Text & " " & Yearfield.Text &
            ", " & Classgroupfield.Text
            Classespercentageschart.Series.Add(series)
            'adds a cumulative series of datapoints
            Classespercentageschart.Series(series).Points.AddY(((fivegrade) /
            dt.Rows.Count))
            Classespercentageschart.Series(series).Points.AddY(((fourgrade + fivegrade) /
            dt.Rows.Count)))
            Classespercentageschart.Series(series).Points.AddY(((threegrade + fourgrade +
            fivegrade) / dt.Rows.Count)))
        Else

```

```

        series = "Class " & p + 1 & "; " & SubjectField.Text & " " & Yearfield.Text &
", " & Classgroupfield.Text
        Classespercentageschart.Series.Add(series)
        'adds a non-cumulative series of datapoints
        Classespercentageschart.Series(series).Points.AddY(((fivegrade) /
dt.Rows.Count))
        Classespercentageschart.Series(series).Points.AddY(((fourgrade) /
dt.Rows.Count))
        Classespercentageschart.Series(series).Points.AddY(((threegrade) /
dt.Rows.Count))
    End If
    'sets the datapoint labels of the graph
    Classespercentageschart.Series(series).Points(0).AxisLabel = "3"
    Classespercentageschart.Series(series).Points(1).AxisLabel = "4"
    Classespercentageschart.Series(series).Points(2).AxisLabel = "5"
End Sub
'this sub deals with A2 Predictions percentages
Public Sub A2Predictions(ByVal sql2 As String)
    'stores subjects
    Dim dt As New DataTable
    'stores students
    Dim dt1 As New DataTable
    'stores studentsgrades
    Dim dt2 As New DataTable
    'stores the grid that will be displayed
    Dim dt3 As New DataTable
    Dim ds As DataSet
    'stores index variables used in my for next statements
    Dim i As Integer
    Dim s As Integer

    'stores the predictedvalue point score of a Grade
    Dim predictedvalue As Double
    'stores the total of all the predictedvalues
    Dim currentpredictedtotal As Double = 0
    'stores the actual predicted grade
    Dim predictedgrade As String
    'stores a subject
    Dim subject As String
    'stores a weighting variable that is applied to the predictedvalue depending on
the subject
    Dim GCSEgradeweighting As Double
    'stores the averagepointscore for a student
    Dim averagepointscore As Double
    'stores the ID of a student
    Dim StudentID As Integer

    'queries all the subject the class is a part of and places it into a datatable
    sql = "SELECT DISTINCT Subject FROM Subject Where Subject.Subject = '" &
SubjectField.Text & "' ORDER BY Subject ASC"
    'the dataset is set after passing the following SQL statement and table into the
GenerateDataset sub of the public module
    ds = generateDataset(sql, "temp")
    'the dt datatable is filled with the a list of subjects from the database
    dt = ds.Tables(0)

    'the dataset is set after passing the SQL2 statement which is set depending on the
user's filter selection, and table into the GenerateDataset sub of the public module
    ds = generateDataset(sql2, "temp")
    'the dt1 table is filled with a list of students
    dt1 = ds.Tables(0)

```

```

'this for next loop loops through every student
'for eachs student the dt2 datatable is filled with that students grades
dt3.Columns.Add("Grade")
For i = 0 To (dt1.Rows.Count - 1)
    averagepointscore = 0
    currentpredictedtotal = 0
    StudentID = dt1.Rows(i)("StudentID").ToString()
    sql = "SELECT DISTINCT Grade.GradeID, Student.Firstname, Student.Surname,
Subject.Subject, Grade.Grade, [Level].[Level], GradeType.GradeType, DatasetID.Year FROM
StudentGroup INNER JOIN Class ON StudentGroup.ClassID = Class.ClassID INNER JOIN Grade
INNER JOIN[Level] ON Grade.LevelID = [Level].LevelID INNER JOIN GradeType ON
Grade.GradeTypeID = GradeType.GradeTypeID INNER JOIN Student ON Grade.StudentID =
Student.StudentID ON StudentGroup.StudentID = Student.StudentID INNER JOIN Subject ON
Grade.SubjectID = Subject.SubjectID INNER JOIN DatasetID ON Grade.DatasetID =
DatasetID.DatasetID WHERE Student.StudentId = '" & StudentID & "' ORDER BY GradeID ASC"
ds = generateDataset(sql, "temp")
dt2 = ds.Tables(0)

'this variable is used to count the number of GCSE grades a student has
Dim count As Integer
count = 0

'this for next statement then loops through all of a students grades and then
applies a predictedvalue depending on the grade, this grade later has a weighting applied
to it
For s = 0 To (dt2.Rows.Count - 1)

    'if the value of the grade underneath the Grade header in the datatable on
    the row of the i index is an A* then this if statement runs
    If dt2.Rows(s)("Grade").ToString() = "A*" Then
        'the GCSE gradeweighting variable is recieved after passing the
        datatable and the row index into the GCSEgradeweighting function in the public module
        GCSEgradeweighting = GCSEGradeweighting(s, dt2)
        'the predicted value is set as a national value but then is multiplied
        by the weighting value
        predictedvalue = 58 * GCSEgradeweighting
        count = count + 1
    End If
    'all of the following if statements work like the one above for all the
    other different grades
    If dt2.Rows(s)("Grade").ToString() = "A" Then
        GCSEgradeweighting = GCSEGradeweighting(s, dt2)
        predictedvalue = 52 * GCSEgradeweighting
        count = count + 1
    End If
    If dt2.Rows(s)("Grade").ToString() = "B" Then
        GCSEgradeweighting = GCSEGradeweighting(s, dt2)
        predictedvalue = 46 * GCSEgradeweighting
        count = count + 1
    End If
    If dt2.Rows(s)("Grade").ToString() = "C" Then
        GCSEgradeweighting = GCSEGradeweighting(s, dt2)
        predictedvalue = 40 * GCSEgradeweighting
        count = count + 1
    End If
    If dt2.Rows(s)("Grade").ToString() = "D" Then
        GCSEgradeweighting = GCSEGradeweighting(s, dt2)
        predictedvalue = 34 * GCSEgradeweighting
        count = count + 1
    End If
    If dt2.Rows(s)("Grade").ToString() = "E" Then

```

```

        GCSEgradeweighting = GCSEGradeweighting(s, dt2)
        predictedvalue = 28 * GCSEgradeweighting
        count = count + 1
    End If
    If dt2.Rows(s)("Grade").ToString() = "F" Then
        GCSEgradeweighting = GCSEGradeweighting(s, dt2)
        predictedvalue = 22 * GCSEgradeweighting
        count = count + 1
    End If
    If dt2.Rows(s)("Grade").ToString() = "G" Then
        GCSEgradeweighting = GCSEGradeweighting(s, dt2)
        predictedvalue = 16 * GCSEgradeweighting
        count = count + 1
    End If
    If dt2.Rows(s)("Grade").ToString() = "4" Then
        GCSEgradeweighting = GCSEGradeweighting(s, dt2)
        predictedvalue = 0
    End If
    If dt2.Rows(s)("Grade").ToString() = "5" Then
        GCSEgradeweighting = GCSEGradeweighting(s, dt2)
        predictedvalue = 0
    End If
    'the current predicted total equals the previous currentpredicted total +
the predicted value
    currentpredictedtotal = currentpredictedtotal + predictedvalue
Next

'a mean average point score for the student is created by taking the total of
all the predicted values and dividing by the number of grades
averagepointscore = currentpredictedtotal / count

'the subject variable is set as the value of the subject field
subject = SubjectField.Text
'the actual predicted grade is set by passing the subject, row index,
datatable and average point score to the A2predictedgrade sub in the public module
predictedgrade = A2predictedgrade(subject, 0, dt, averagepointscore)
'the predicted grade is then written on the row index of i underneath the
grade header
dt3.Rows.Add()
dt3.Rows(i)("Grade") = predictedgrade
Next

'this for next statement loops itself to the last row containing a grade
'it counts the number of each grade within the dataset
For i = 0 To (dt3.Rows.Count - 1)
    'if there is a 1 under any of the columns in the dataset for each row then the
count of that corresponding grade increases by 1

    If dt3.Rows(i)("Grade").ToString() = "A*" Then
        Astargrade = Astargrade + 1
    End If
    If dt3.Rows(i)("Grade").ToString() = "A" Then
        Agrade = Agrade + 1
    End If
    If dt3.Rows(i)("Grade").ToString() = "B" Then
        Bgrade = Bgrade + 1
    End If
    If dt3.Rows(i)("Grade").ToString() = "C" Then
        Cgrade = Cgrade + 1
    End If
    If dt3.Rows(i)("Grade").ToString() = "D" Then
        Dgrade = Dgrade + 1
    End If

```

```

        End If
        If dt3.Rows(i)("Grade").ToString() = "E" Then
            Egrade = Egrade + 1
        End If
        If dt3.Rows(i)("Grade").ToString() = "F" Then
            Fgrade = Fgrade + 1
        End If
    Next

    'sets the value of the text labels within the table that corresponds to the index
    value of p
    If p = 0 Then
        'the text values of the labels in the table when the tableviewbutton is
        clicked are set to their corresponding number of grades counted from the dataset
        Utable1.Text = "U"
        tbl1g.Visible = False
        tbl1ggrade.Visible = False
        tbl1astar.Text = Astargrade & " out of " & dt3.Rows.Count
        tbl1a.Text = Agrade & " out of " & dt3.Rows.Count
        tbl1b.Text = Bgrade & " out of " & dt3.Rows.Count
        tbl1c.Text = Cgrade & " out of " & dt3.Rows.Count
        tbl1d.Text = Dgrade & " out of " & dt3.Rows.Count
        tbl1e.Text = Egrade & " out of " & dt3.Rows.Count
        tbl1f.Text = Fgrade & " out of " & dt3.Rows.Count
        'this sets the label of the table, so the user knows what the data in the
        table correlates to
        Table1label.Text = "Subject " & p + 1 & ":" & SubjectField.Text & " A2
Predicts " & vbCrLf & Yearfield.Text & ", " & Classgroupfield.Text
        'this sets the atoc label text of the table by adding up all the grades from
        A* to c and then dividing by the total number of grades in the dataset, which is then
        converted to a percentage and displayed as a percentage
        tbl1atoc.Text = (Agrade + Bgrade + Cgrade + Astargrade) & " out of " &
dt3.Rows.Count & " (~" & Int(((Agrade + Bgrade + Cgrade + Astargrade) / dt3.Rows.Count) *
100) & "%)"
    End If
    If p = 1 Then
        Utable2.Text = "U"
        tbl2g.Visible = False
        tbl2ggrade.Visible = False
        tbl2astar.Text = Astargrade & " out of " & dt3.Rows.Count
        tbl2a.Text = Agrade & " out of " & dt3.Rows.Count
        tbl2b.Text = Bgrade & " out of " & dt3.Rows.Count
        tbl2c.Text = Cgrade & " out of " & dt3.Rows.Count
        tbl2d.Text = Dgrade & " out of " & dt3.Rows.Count
        tbl2e.Text = Egrade & " out of " & dt3.Rows.Count
        tbl2f.Text = Fgrade & " out of " & dt3.Rows.Count
        Table2label.Text = "Subject " & p + 1 & ":" & SubjectField.Text & " A2
Predicts " & vbCrLf & Yearfield.Text & ", " & Classgroupfield.Text
        tbl2atoc.Text = (Agrade + Bgrade + Cgrade + Astargrade) & " out of " &
dt3.Rows.Count & " (~" & Int(((Agrade + Bgrade + Cgrade + Astargrade) / dt3.Rows.Count) *
100) & "%)"
    End If
    If p = 2 Then
        Utable3.Text = "U"
        tbl3g.Visible = False
        tbl3ggrade.Visible = False
        tbl3astar.Text = Astargrade & " out of " & dt3.Rows.Count
        tbl3a.Text = Agrade & " out of " & dt3.Rows.Count
        tbl3b.Text = Bgrade & " out of " & dt3.Rows.Count
        tbl3c.Text = Cgrade & " out of " & dt3.Rows.Count
        tbl3d.Text = Dgrade & " out of " & dt3.Rows.Count
        tbl3e.Text = Egrade & " out of " & dt3.Rows.Count

```

```

tbl3f.Text = Fgrade & " out of " & dt3.Rows.Count
Table3label.Text = "Subject " & p + 1 & ":" & SubjectField.Text & " A2
Predicts " & vbCrLf & Yearfield.Text & ", " & Classgroupfield.Text
tbl3atoc.Text = (Agrade + Bgrade + Cgrade + Astargrade) & " out of " &
dt3.Rows.Count & " (~" & Int(((Agrade + Bgrade + Cgrade + Astargrade) / dt3.Rows.Count) *
100) & "%)"
End If
If p = 3 Then
tbl4g.Visible = False
tbl4ggrade.Visible = False
Utable4.Text = "U"
tbl4astar.Text = Astargrade & " out of " & dt3.Rows.Count
tbl4a.Text = Agrade & " out of " & dt3.Rows.Count
tbl4b.Text = Bgrade & " out of " & dt3.Rows.Count
tbl4c.Text = Cgrade & " out of " & dt3.Rows.Count
tbl4d.Text = Dgrade & " out of " & dt3.Rows.Count
tbl4e.Text = Egrade & " out of " & dt3.Rows.Count
tbl4f.Text = Fgrade & " out of " & dt3.Rows.Count
Table4label.Text = "Subject " & p + 1 & ":" & SubjectField.Text & " A2
Predicts " & vbCrLf & Yearfield.Text & ", " & Classgroupfield.Text
tbl4atoc.Text = (Agrade + Bgrade + Cgrade + Astargrade) & " out of " &
dt3.Rows.Count & " (~" & Int(((Agrade + Bgrade + Cgrade + Astargrade) / dt3.Rows.Count) *
100) & "%)"
End If

'this if statement selects what type of graph to display based on the user's
selection
If Graphtypefield.Text = "Cumulative Grade Percentage Graph" Then
    'name of the series is set dynamically depending on users choices
    series = "Subject " & p + 1 & ":" & SubjectField.Text & " A2 Predicted " &
Yearfield.Text & ", " & Classgroupfield.Text
    'adds a series to the graph
    Classespercentageschart.Series.Add(series)
    'adds datapoints to the series of the graph, these points work cumulatively so
    that each following data point adds all the previous data points values to it
    Classespercentageschart.Series(series).Points.AddY(Astargrade /
dt3.Rows.Count)
    Classespercentageschart.Series(series).Points.AddY((Agrade + Astargrade) /
dt3.Rows.Count)
    Classespercentageschart.Series(series).Points.AddY((Bgrade + Astargrade +
Agrade) / dt3.Rows.Count)
    Classespercentageschart.Series(series).Points.AddY((Cgrade + Astargrade +
Agrade + Bgrade) / dt3.Rows.Count)
    Classespercentageschart.Series(series).Points.AddY((Dgrade + Astargrade +
Agrade + Bgrade + Cgrade) / dt3.Rows.Count)
    Classespercentageschart.Series(series).Points.AddY((Egrade + Dgrade +
Astargrade + Agrade + Bgrade + Cgrade) / dt3.Rows.Count)
    Classespercentageschart.Series(series).Points.AddY((Fgrade + Egrade + Dgrade +
Astargrade + Agrade + Bgrade + Cgrade) / dt3.Rows.Count)
Else
    'name of the series is set dynamically depending on users choices
    series = "Subject " & p + 1 & ":" & SubjectField.Text & " A2 Predicted " &
Yearfield.Text & ", " & Classgroupfield.Text
    'adds a series to the graph
    Classespercentageschart.Series.Add(series)
    'adds datapoints to the series of the graph equivalent to the number of grades
    counted
    Classespercentageschart.Series(series).Points.AddY(Astargrade /
dt3.Rows.Count)
    Classespercentageschart.Series(series).Points.AddY(Agrade / dt3.Rows.Count)
    Classespercentageschart.Series(series).Points.AddY(Bgrade / dt3.Rows.Count)
    Classespercentageschart.Series(series).Points.AddY(Cgrade / dt3.Rows.Count)

```

```

Classespercentageschart.Series(series).Points.AddY(Dgrade / dt3.Rows.Count)
Classespercentageschart.Series(series).Points.AddY(Egrade / dt3.Rows.Count)
Classespercentageschart.Series(series).Points.AddY(Fgrade / dt3.Rows.Count)
End If

'sets the labels of the datapoints on the series
Classespercentageschart.Series(series).Points(0).AxisLabel = "A*"
Classespercentageschart.Series(series).Points(1).AxisLabel = "A"
Classespercentageschart.Series(series).Points(2).AxisLabel = "B"
Classespercentageschart.Series(series).Points(3).AxisLabel = "C"
Classespercentageschart.Series(series).Points(4).AxisLabel = "D"
Classespercentageschart.Series(series).Points(5).AxisLabel = "E"
Classespercentageschart.Series(series).Points(6).AxisLabel = "U"
End Sub
'this sub runs when the view percentages button is clicked by the user
Private Sub Addsearch_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Addsearch.Click
    'if the state of the txtvalidate function is true then the following code runs
    If textvalidate() = True Then
        'this if statement validates the users selection, since only english language
        or maths are stored for KS2, the program prevents the user from displaying an empty graph
        for KS2 if any other subject is chosen. An appropriate error message is also displayed
        If Levelfield.Text = "KS2" And SubjectField.Text <> "English Language" And
SubjectField.Text <> "Maths" Then
            MsgBox("That subject is not a part of that KeyStage(Level)",
MsgBoxStyle.Information, MessageBoxButtons.OK)
        Else
            'prevents the user from adding more than 4 series
            If p = 4 Then
                MsgBox("You cannot compare any more than 4 Chart series")
            Else
                'stores a dataset
                Dim ds As New DataSet
                'stores a datatable
                Dim dt As New DataTable

                'the count of the grade variables are reset to 0
                Astartgrade = 0
                Agrade = 0
                Bgrade = 0
                Cgrade = 0
                Dgrade = 0
                Egrade = 0
                Fgrade = 0
                Ggrade = 0
                threegrade = 0
                fourgrade = 0
                fivegrade = 0

                'the current graph series and title labels are cleared if they have
already been set
                Classespercentageschart.Titles.Clear()
                'the toggle labels button becomes visible
                labelstoggle.Visible = True

                'calls the Loadgraphic class
                Dim objPlsWait As New clsOPAWaitScreen
                objPlsWait.ShowWaitScreen("")

                If Gradetypefield.Text = "A2 Predicted" Then
                    tablettitle1.Text = "Number of Student's Predicted Grade"

```

```

        tablettitle2.Text = "Number of Student's Predicted Grade"
        tablettitle3.Text = "Number of Student's Predicted Grade"
        tablettitle4.Text = "Number of Student's Predicted Grade"

        'sets the chart label text depending on the users graphtype
selection
        If Graphtypefield.Text = "Cumulative Grade Percentage Graph" Then
            chartlabel.Text = "Cumulative Predictions by Grade (%)"
        Else
            chartlabel.Text = "Percentage of Students Predicted a Grade
(%)"
        End If

        'selects all students within the year the user has selected in the
yearfield
        Dim sql2 As String
        sql2 = "SELECT DISTINCT Student.StudentID, Student.Firstname,
Student.Surname FROM StudentGroup INNER JOIN Class ON StudentGroup.ClassID = Class.ClassID
INNER JOIN Grade INNER JOIN[Level] ON Grade.LevelID = [Level].LevelID INNER JOIN GradeType
ON Grade.GradeTypeID = GradeType.GradeTypeID INNER JOIN Student ON Grade.StudentID =
Student.StudentID ON StudentGroup.StudentID = Student.StudentID INNER JOIN Subject ON
Grade.SubjectID = Subject.SubjectID INNER JOIN DatasetID ON Grade.DatasetID =
DatasetID.DatasetID WHERE Student.DatasetID = '" & Yearfield.SelectedValue & "' AND
Class.Classgroup = '" & Classgroupfield.Text & "' ORDER BY Student.StudentID ASC"
            'passes the SQL statement to the A2predictions sub
A2Predictions(sql2)

        Else
            'sets the chart label text depending on the users graphtype
selection
            If Graphtypefield.Text = "Cumulative Grade Percentage Graph" Then
                chartlabel.Text = "Cumulative Percentage by Grade (%)"
            Else
                chartlabel.Text = "Percentage of Students Achieving a Grade
(%)"
            End If

            If Levelfield.Text = "KS4" Then
                'if the levelfield value is KS4 then this code runs
                'runs the KS4 percentages sub
                KS4percentages()
                'the graph becomes visible
                Classespercentageschart.Visible = True
            Else
                'runs the KS2 percentages sub
                KS2percentages()
                'the graph becomes visible
                Classespercentageschart.Visible = True
            End If
        End If

        'disables the legend from the graph
        Classespercentageschart.Series(p).IsVisibleInLegend = True
        'shows the values of datapoints
        Classespercentageschart.Series(p).IsValueShownAsLabel = True
        'sets the colour of the datapoint labels as maroon
        Classespercentageschart.Series(p).ChartType =
DataVisualization.Charting.SeriesChartType.Line
        'changes the chart colour to skyblue
        Classespercentageschart.Series(p).LabelForeColor = Color.Maroon
    
```

```

    'changes the chart to a 3D bar style
    Classespercentageschart.Series(0).Color = Color.DarkGreen
    'sets the minimum and maximum axis
    Classespercentageschart.ChartAreas(0).AxisY.Minimum = 0
    Classespercentageschart.ChartAreas(0).AxisY.Maximum = 1
    'sets the format of the chart labels
    Classespercentageschart.ChartAreas(0).AxisY.LabelStyle.Format = "#%"
    'sets the format of the series labels
    Classespercentageschart.Series(p).LabelFormat = "##.##" & "%"
    'sets the colour of the axis labels
    Classespercentageschart.ChartAreas(0).AxisX.LabelStyle.ForeColor =
Color.Maroon
    Classespercentageschart.ChartAreas(0).AxisY.LabelStyle.ForeColor =
Color.Maroon

    'sets the background of the chart as transparent
    Classespercentageschart.ChartAreas(0).BackColor = Color.Transparent
    'increments the value of p by 1 as a new series is added by the user
    p = p + 1
    'graph becomes visible
    Classespercentageschart.Visible = True
    'if the user selects KS4 then the groupview groupbox is visible
    If Levelfield.Text = "KS4" Then
        Groupview.Visible = True
    End If

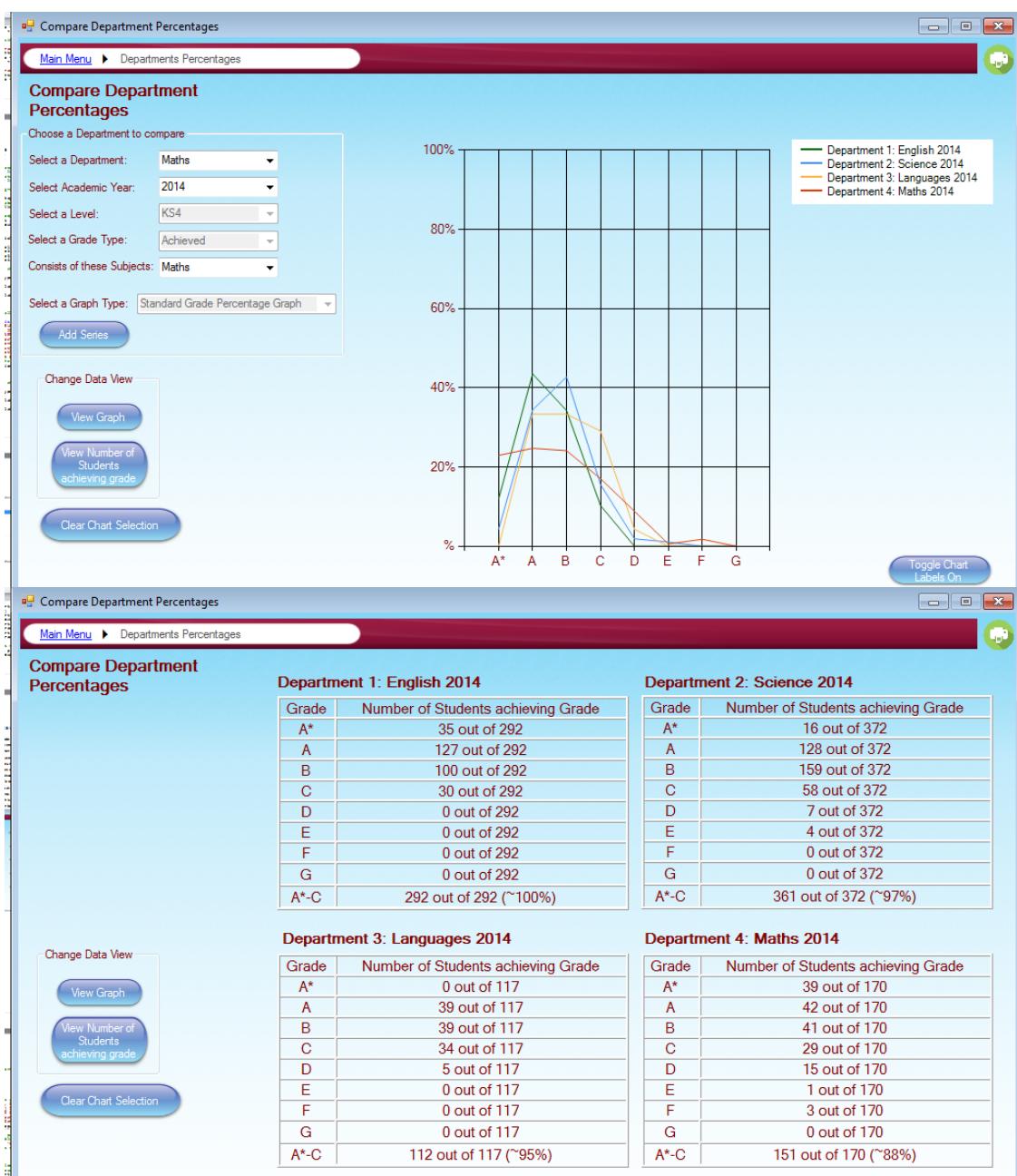
    'these fields are no longer enables so that the user cannot add a
    series for KS2 and KS4, or add a series ofr one graph type and a series for the other
    graph type
    Graphtypefield.Enabled = False
    Levelfield.Enabled = False
    Gradetypefield.Enabled = False

    'closes the loadgraphic wait and resets its objplswait variable
    objPlsWait.CloseWaitScreen()
    objPlsWait = Nothing
    Addsearch.Text = "Add Series"
    End If
End If
Else
    'if the state of the txtvalidate function is not true then the following error
    message displays and nothing else happens
    MsgBox("One or more textboxes have not been completed",
    MsgBoxStyle.Information, MessageBoxButtons.OK)
    End If
End Sub
'deals with the print form button
Private Sub printbutton_Click(sender As System.Object, e As System.EventArgs) Handles
printbutton.Click
    'sets the printer settings to default
    PrintForm.PrinterSettings = PrintDialog.PrinterSettings
    If PrintDialog.ShowDialog() = DialogResult.OK Then
        'allows the user to change print settings
        PrintForm.PrinterSettings = PrintDialog.PrinterSettings
        'changes the orientation of the printed document to landscape
        Me.PrintForm.PrinterSettings.DefaultPageSettings.Landscape = True
        'refreshes the form so that the printdialog is not printed along with the form
        Me.Refresh()
        'prints the current form
        Me.PrintForm.Print()
    End If
End Sub
End Class

```


Compare Department Percentages

This form allows a user to compare grade statistics of achieved GCSE grades or predicted A2 grades. This is done by comparing whole departments (consisting of all their respective subjects) within years, a user can select which department they want to compare and which grade type and level they want to compare. For example a user could compare the achieved GCSE statistics for the humanities department for four consecutive years. A maximum of four departments can be compared at any one time. The statistics can be displayed as percentages on a linear line graph or a cumulative grade line graph. There is also a viewing mode that allows the user to view the raw statistics of students achieving or predicted individual grades and the number of students achieving or predicted A* - C grades. At any time the current departments that are being compared can be cleared so that a new set of departments can be compared. A printed copy of this form can be produced by using the print icon, which is very useful as a copy of the raw statistics can be produced or a copy of the graph comparisons can be produced.



```

Addsearch System.Windows.Forms.Button
banner System.Windows.Forms.PictureBox
chartlabel System.Windows.Forms.Label
Classlabel System.Windows.Forms.Label
Clearselection System.Windows.Forms.Button
Compare_Department_Percentages System.Windows.Forms.Form
Comparedepartmentgroup System.Windows.Forms.GroupBox
Departmentfield System.Windows.Forms.ComboBox
Departmentpercentageschart System.Windows.Forms.DataVisualization.Charting.Chart
departmentspercentageslabel System.Windows.Forms.LinkLabel
Gradetypfield System.Windows.Forms.ComboBox
Graphtypfield System.Windows.Forms.ComboBox
Graphview System.Windows.Forms.Button
labelstoggle System.Windows.Forms.Button
Levelfield System.Windows.Forms.ComboBox
Levelelabel System.Windows.Forms.Label
MainMenu System.Windows.Forms.LinkLabel
printbutton System.Windows.Forms.Button
PrintDialog System.Windows.Forms.PrintDialog
PrintForm Microsoft.VisualBasic.PowerPacks.Printing.PrintForm
Seriestable1 System.Windows.Forms.TableLayoutPanel
Seriestable2 System.Windows.Forms.TableLayoutPanel
Seriestable3 System.Windows.Forms.TableLayoutPanel
Seriestable4 System.Windows.Forms.TableLayoutPanel
SubjectField System.Windows.Forms.ComboBox
subjectlabel System.Windows.Forms.Label
Table1label System.Windows.Forms.Label
Table2label System.Windows.Forms.Label
Table3label System.Windows.Forms.Label
Table4label System.Windows.Forms.Label
TablesView System.Windows.Forms.Button
tabletitle1 System.Windows.Forms.Label
tabletitle2 System.Windows.Forms.Label
tabletitle3 System.Windows.Forms.Label
tabletitle4 System.Windows.Forms.Label
tbl1a System.Windows.Forms.Label
tbl1astar System.Windows.Forms.Label
tbl1atoc System.Windows.Forms.Label
tbl1b System.Windows.Forms.Label
tbl1c System.Windows.Forms.Label
tbl1d System.Windows.Forms.Label
tbl1e System.Windows.Forms.Label
tbl1f System.Windows.Forms.Label

```

```

tbl1c System.Windows.Forms.Label
tbl1d System.Windows.Forms.Label
tbl1e System.Windows.Forms.Label
tbl1f System.Windows.Forms.Label
tbl2a System.Windows.Forms.Label
tbl2astar System.Windows.Forms.Label
tbl2atoc System.Windows.Forms.Label
tbl2b System.Windows.Forms.Label
tbl2c System.Windows.Forms.Label
tbl2d System.Windows.Forms.Label
tbl2e System.Windows.Forms.Label
tbl2f System.Windows.Forms.Label
tbl3a System.Windows.Forms.Label
tbl3astar System.Windows.Forms.Label
tbl3atoc System.Windows.Forms.Label
tbl3b System.Windows.Forms.Label
tbl3c System.Windows.Forms.Label
tbl3d System.Windows.Forms.Label
tbl3e System.Windows.Forms.Label
tbl3f System.Windows.Forms.Label
tbl4a System.Windows.Forms.Label
tbl4astar System.Windows.Forms.Label
tbl4atoc System.Windows.Forms.Label
tbl4b System.Windows.Forms.Label
tbl4c System.Windows.Forms.Label
tbl4d System.Windows.Forms.Label
tbl4e System.Windows.Forms.Label
tbl4f System.Windows.Forms.Label
Yearfield System.Windows.Forms.ComboBox
yearlabel System.Windows.Forms.Label

```

Parameter Name	Description
CompareDepartment_Percentages	This sub connects to the database server when the form initially loads and dynamically generates the list properties of all the combo boxes on the form and sets their value members. It also sets the default text property of the graph type combo box.
MainMenu_LinkClicked	When the main menu link label is clicked all open forms close and the main menu displays.
textvalidate	This function sets a true or false condition depending on whether any textboxes on the form are blank. It also sets the background colour of textboxes appropriately to indicate to the user that fields have not been completed.

ks4percentages	Selects all grades within the database and then counts the number of each KS4 grade e.g. A's B's. It then sets the text properties of the values within the statistics table and adds data points to the series of the chart, these data points depend on the user's selection of a standard or cumulative graph type. It then adds the corresponding data point axis labels for each KS4 grade.
KS2percentages	Selects all grades within the database and then counts the number of each KS2 grade e.g. 3's 4's. It then sets the text properties of the values within the statistics table and adds data points to the series of the chart, these data points depend on the user's selection of a standard or cumulative graph type. It then adds the corresponding data point axis labels for each KS2 grade.
A2Predictions	This sub generates a series of predicted grades by first selecting all students that match the filter criteria, and calculating an average point score for each student based on their GCSE grades. From this average point score a predicted grade is then generated for the selected department the user want to view predicted grades for. Once all of the predicted grades have been generated, data points are added to the chart series, these are either linear data points or cumulative data points depending on the users graph choice. The label properties of the statistics table are also set as the number of each predicted grade. Corresponding data point axis labels for each predicted grade are also added to the chart.
Addsearch_Click	When the add series button is clicked by the user, this sub determines which type of series to add to the chart. It is also responsible for validation and formatting the chart.
Clearselection_Click	Creates a new instance of every object on the form (essentially re-loads the form).
labelstoggle_Click	Handles the button click event of the Toggle chart labels button. When clicked an index variable loops through all series and changes the visible property of the labels to false. It also changes the text property of the button.
Graphview_Click	When the view graph button is clicked all objects associated with the graph visible to the user and all objects associated with the statistics tables non-visible to the user.
TablesView_Click	When the view table's button is clicked all objects associated with the graph become non-visible to the user and all objects associated with

	the statistics tables become visible to the user.
printbutton_Click	This sub allows the user to print a hard copy of the form when the print icon is clicked, additional print dialog options are then displayed allowing the user to choose printer and number of copies amongst other settings.
DepartmentField_SelectedIndexChanged	This sub handles every instance the department field is altered by the user's selection. It validates whether all of the required text fields have been completed and if so runs the subjectsfieldgenerate sub.
Levelfield_SelectedIndexChanged	Whenever the text value of the level field is changed this sub determines whether the grade type field is either enabled or not enabled.
Yearfield_SelectedIndexChanged	This sub handles every instance the year field is altered by the user's selection. It validates whether all of the required text fields have been completed and if so runs the subjectsfieldgenerate sub.
subjectsfieldgenerate	This sub queries the database using the user's selected department filters to return all the subjects that are a part of the department the user has selected. The returned results are then set as the list property of the subjectfield combo box.

```

Imports the following system code commands so that I can use them appropriately
Imports System.Data.SqlClient
Imports System.Data
Imports System.Configuration
Imports System.Data.DataTable
Public Class Compare_Department_Percentages
    'stores series as an empty string that can be used within this class
    Dim series As String = Nothing
    'i and p are used as an index variable in my for next statements
    Dim i As Integer = 0
    Dim p As Integer = 0
    'these variables store the number of grades counted
    Dim threegrade As Integer = 0
    Dim fourgrade As Integer = 0
    Dim fivegrade As Integer = 0
    Dim Astargrade As Integer
    Dim Agrade As Integer = 0
    Dim Bgrade As Integer = 0
    Dim Cgrade As Integer = 0
    Dim Dgrade As Integer = 0
    Dim Egrade As Integer = 0
    Dim Fgrade As Integer = 0
    Dim Ggrade As Integer = 0
    'this sub runs when the form initially loads
    Private Sub CompareDepartment_Percentages(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Me.Load
        'runs the databaseconnect sub from the public module
        DatabaseConnect()
        'stores dt as datatable
        Dim dt As DataTable
        p = 0

        'Uses the Generate combo function of the module to populate a combobox by passing
        the table, field and needempty variables to the Generatecombo function and then returning
        the result in a datatable
        dt = GenerateCombo("DatasetID", "DatasetID, Year", True)
        'sets the datasource of the combobox
        Yearfield.DataSource = dt
        'sets the display member of the combobox
        Yearfield.DisplayMember = "Year"
        'the index of each item in the combobox is set as their corresponding ID from the
        database
        Yearfield.ValueMember = "DatasetID"

        dt = GenerateCombo("Department", "DepartmentID, Department", True)
        Departmentfield.DataSource = dt
        Departmentfield.DisplayMember = "Department"
        Departmentfield.ValueMember = "DepartmentID"

        dt = GenerateCombo("Level", "levelID, Level", True)
        Levelfield.DataSource = dt
        Levelfield.DisplayMember = "Level"
        Levelfield.ValueMember = "LevelID"

        dt = GenerateCombo("GradeType", "GradeTypeID, GradeType", True)
        Gradetypefield.DataSource = dt
        Gradetypefield.DisplayMember = "GradeType"
        Gradetypefield.ValueMember = "GradeTypeID"

        'sets the default text of the graphytypefield
        Graphtypefield.Text = "Standard Grade Percentage Graph"

```

```

End Sub
'this sub deals with KS4 percentages
Public Sub KS4percentages()
    'stores ds as a dataset
    Dim ds As New DataSet
    'stores dt as a datatable
    Dim dt As New DataTable

    ' This SQL statement fills a dataset depending on the users department choice
    sql = "SELECT Grade.Grade, Grade.DatasetID, Grade.LevelID, Grade.SubjectID FROM
Department INNER JOIN [Level] INNER JOIN Grade ON [Level].LevelID = Grade.LevelID INNER
JOIN DatasetID ON Grade.DatasetID = DatasetID.DatasetID INNER JOIN Subject ON
Grade.SubjectID = Subject.SubjectID ON Department.DepartmentID = Subject.DepartmentID
WHERE Grade.DatasetID = '" & Yearfield.SelectedValue & "' and Department.DepartmentID =
'" & Departmentfield.SelectedValue & "' and Grade.LevelID = '" & Levelfield.SelectedValue
& "'"
    'the dataset is set after passing the following SQL statement and table into the
GenerateDataset sub of the public module
    ds = generateDataset(sql, "temp")
    'the datatable contents is set as the returned dataset once the SQL statement has
filled the dataset
    dt = ds.Tables(0)

    'this for next statement loops itself to the last row containing a grade
    'it counts the number of each grade within the dataset
    For i = 0 To (dt.Rows.Count - 1)
        'if there is a matching grade under any of the columns in the dataset for each
        row then the count of that corresponding grade count increases by 1
        If dt.Rows(i)("Grade").ToString() = "A*" Then
            Astargrade = Astargrade + 1
        End If
        If dt.Rows(i)("Grade").ToString() = "A" Then
            Agrade = Agrade + 1
        End If
        If dt.Rows(i)("Grade").ToString() = "B" Then
            Bgrade = Bgrade + 1
        End If
        If dt.Rows(i)("Grade").ToString() = "C" Then
            Cgrade = Cgrade + 1
        End If
        If dt.Rows(i)("Grade").ToString() = "D" Then
            Dgrade = Dgrade + 1
        End If
        If dt.Rows(i)("Grade").ToString() = "E" Then
            Egrade = Egrade + 1
        End If
        If dt.Rows(i)("Grade").ToString() = "F" Then
            Fgrade = Fgrade + 1
        End If
        If dt.Rows(i)("Grade").ToString() = "G" Then
            Ggrade = Ggrade + 1
        End If
    Next

    'sets the value of the text labels within the table that corresponds to the index
    value of p
    If p = 0 Then
        'the text values of the labels in the table when the tableviewbutton is
        clicked are set to their corresponding number of grades counted from the dataset
        tbl1astar.Text = Astargrade & " out of " & dt.Rows.Count
        tbl1a.Text = Agrade & " out of " & dt.Rows.Count
        tbl1b.Text = Bgrade & " out of " & dt.Rows.Count

```

```

tbl1c.Text = Cgrade & " out of " & dt.Rows.Count
tbl1d.Text = Dgrade & " out of " & dt.Rows.Count
tbl1e.Text = Egrade & " out of " & dt.Rows.Count
tbl1f.Text = Fgrade & " out of " & dt.Rows.Count
tbl1ggrade.Text = Ggrade & " out of " & dt.Rows.Count
'this sets the label of the table, so the user knows what the data in the
table correlates to
Table1label.Text = "Department " & p + 1 & ":" & Departmentfield.Text & " " &
Yearfield.Text
'this sets the atoc label text of the table by adding up all the grades from
A* to c and then dividing by the total number of grades in the dataset, which is then
converted to a percentage and displayed as a percentage
tbl1atoc.Text = (Agrade + Bgrade + Cgrade + Astargrade) & " out of " &
dt.Rows.Count & " (~" & Int(((Agrade + Bgrade + Cgrade + Astargrade) / dt.Rows.Count) *
100) & "%)"
End If
If p = 1 Then
    tbl2astar.Text = Astargrade & " out of " & dt.Rows.Count
    tbl2a.Text = Agrade & " out of " & dt.Rows.Count
    tbl2b.Text = Bgrade & " out of " & dt.Rows.Count
    tbl2c.Text = Cgrade & " out of " & dt.Rows.Count
    tbl2d.Text = Dgrade & " out of " & dt.Rows.Count
    tbl2e.Text = Egrade & " out of " & dt.Rows.Count
    tbl2ggrade.Text = Ggrade & " out of " & dt.Rows.Count
    tbl2f.Text = Fgrade & " out of " & dt.Rows.Count
    Table2label.Text = "Department " & p + 1 & ":" & Departmentfield.Text & " " &
Yearfield.Text
    tbl2atoc.Text = (Agrade + Bgrade + Cgrade + Astargrade) & " out of " &
dt.Rows.Count & " (~" & Int(((Agrade + Bgrade + Cgrade + Astargrade) / dt.Rows.Count) *
100) & "%)"
End If
If p = 2 Then
    tbl3astar.Text = Astargrade & " out of " & dt.Rows.Count
    tbl3a.Text = Agrade & " out of " & dt.Rows.Count
    tbl3b.Text = Bgrade & " out of " & dt.Rows.Count
    tbl3c.Text = Cgrade & " out of " & dt.Rows.Count
    tbl3d.Text = Dgrade & " out of " & dt.Rows.Count
    tbl3e.Text = Egrade & " out of " & dt.Rows.Count
    tbl3ggrade.Text = Ggrade & " out of " & dt.Rows.Count
    tbl3f.Text = Fgrade & " out of " & dt.Rows.Count
    Table3label.Text = "Department " & p + 1 & ":" & Departmentfield.Text & " " &
Yearfield.Text
    tbl3atoc.Text = (Agrade + Bgrade + Cgrade + Astargrade) & " out of " &
dt.Rows.Count & " (~" & Int(((Agrade + Bgrade + Cgrade + Astargrade) / dt.Rows.Count) *
100) & "%)"
End If
If p = 3 Then
    tbl4astar.Text = Astargrade & " out of " & dt.Rows.Count
    tbl4a.Text = Agrade & " out of " & dt.Rows.Count
    tbl4b.Text = Bgrade & " out of " & dt.Rows.Count
    tbl4c.Text = Cgrade & " out of " & dt.Rows.Count
    tbl4d.Text = Dgrade & " out of " & dt.Rows.Count
    tbl4e.Text = Egrade & " out of " & dt.Rows.Count
    tbl4f.Text = Fgrade & " out of " & dt.Rows.Count
    tbl4ggrade.Text = Ggrade & " out of " & dt.Rows.Count
    Table4label.Text = "Department " & p + 1 & ":" & Departmentfield.Text & " " &
Yearfield.Text
    tbl4atoc.Text = (Agrade + Bgrade + Cgrade + Astargrade) & " out of " &
dt.Rows.Count & " (~" & Int(((Agrade + Bgrade + Cgrade + Astargrade) / dt.Rows.Count) *
100) & "%)"
End If

```

```

'this if statement selects what type of graph to display based on the user's
selection
If Graphtypefield.Text = "Cumulative Grade Percentage Graph" Then
    'name of the series is set dynamically depending on users choices
    series = "Department " & p + 1 & ":" & Departmentfield.Text & " " &
Yearfield.Text
    'adds a series to the graph
    Departmentpercentageschart.Series.Add(series)
    'adds datapoints to the series of the graph, these points work cumulatively so
    that each following data point adds all the previous data points values to it
    Departmentpercentageschart.Series(series).Points.AddY(Astargrade /
dt.Rows.Count)
    Departmentpercentageschart.Series(series).Points.AddY((Agrade + Astargrade) /
dt.Rows.Count)
    Departmentpercentageschart.Series(series).Points.AddY((Bgrade + Astargrade +
Agrade) / dt.Rows.Count)
    Departmentpercentageschart.Series(series).Points.AddY((Cgrade + Astargrade +
Agrade + Bgrade) / dt.Rows.Count)
    Departmentpercentageschart.Series(series).Points.AddY((Dgrade + Astargrade +
Agrade + Bgrade + Cgrade) / dt.Rows.Count)
    Departmentpercentageschart.Series(series).Points.AddY((Egrade + Dgrade +
Astargrade + Agrade + Bgrade + Cgrade) / dt.Rows.Count)
    Departmentpercentageschart.Series(series).Points.AddY((Fgrade + Egrade +
Dgrade + Astargrade + Agrade + Bgrade + Cgrade) / dt.Rows.Count)
    Departmentpercentageschart.Series(series).Points.AddY((Ggrade + Fgrade +
Egrade + Dgrade + Astargrade + Agrade + Bgrade + Cgrade) / dt.Rows.Count)
Else
    'name of the series is set dynamically depending on users choices
    series = "Department " & p + 1 & ":" & Departmentfield.Text & " " &
Yearfield.Text
    'adds a series to the graph
    Departmentpercentageschart.Series.Add(series)
    'adds datapoints to the series of the graph equivalent to the number of grades
counted
    Departmentpercentageschart.Series(series).Points.AddY(Astargrade /
dt.Rows.Count)
    Departmentpercentageschart.Series(series).Points.AddY(Agrade / dt.Rows.Count)
    Departmentpercentageschart.Series(series).Points.AddY(Bgrade / dt.Rows.Count)
    Departmentpercentageschart.Series(series).Points.AddY(Cgrade / dt.Rows.Count)
    Departmentpercentageschart.Series(series).Points.AddY(Dgrade / dt.Rows.Count)
    Departmentpercentageschart.Series(series).Points.AddY(Egrade / dt.Rows.Count)
    Departmentpercentageschart.Series(series).Points.AddY(Fgrade / dt.Rows.Count)
    Departmentpercentageschart.Series(series).Points.AddY(Ggrade / dt.Rows.Count)
End If

'sets the labels of the datapoints on the series
Departmentpercentageschart.Series(series).Points(0).AxisLabel = "A*"
Departmentpercentageschart.Series(series).Points(1).AxisLabel = "A"
Departmentpercentageschart.Series(series).Points(2).AxisLabel = "B"
Departmentpercentageschart.Series(series).Points(3).AxisLabel = "C"
Departmentpercentageschart.Series(series).Points(4).AxisLabel = "D"
Departmentpercentageschart.Series(series).Points(5).AxisLabel = "E"
Departmentpercentageschart.Series(series).Points(6).AxisLabel = "F"
Departmentpercentageschart.Series(series).Points(7).AxisLabel = "G"
End Sub
'this sub deals with KS2 percentages and works similarly to the Sub above
Public Sub KS2percentages()
    Dim ds As New DataSet
    Dim dt As New DataTable

    ' This SQL statement fills a dataset depending on the users class choice

```

```

sql = "SELECT Grade.Grade, Grade.DatasetID, Grade.LevelID, Grade.SubjectID FROM
Department INNER JOIN [Level] INNER JOIN Grade ON [Level].LevelID = Grade.LevelID INNER
JOIN DatasetID ON Grade.DatasetID = DatasetID.DatasetID INNER JOIN Subject ON
Grade.SubjectID = Subject.SubjectID ON Department.DepartmentID = Subject.DepartmentID
WHERE Grade.DatasetID = '" & Yearfield.SelectedValue & "' and Department.DepartmentID =
'" & Departmentfield.SelectedValue & "' and Grade.LevelID = '" & Levelfield.SelectedValue
& "'"
ds = generateDataset(sql, "temp")
dt = ds.Tables(0)

'counts the number of KS2 grades
For i = 0 To (dt.Rows.Count - 1)
    If dt.Rows(i)("Grade").ToString() = "3" Then
        threegrade = threegrade + 1
    End If
    If dt.Rows(i)("Grade").ToString() = "4" Then
        fourgrade = fourgrade + 1
    End If
    If dt.Rows(i)("Grade").ToString() = "5" Then
        fivegrade = fivegrade + 1
    End If
Next
'sets the type of series to add to the graph based on whether the user wants to
see a cumulative or non-cumulative graph
If Graphtypefield.Text = "Cumulative Grade Percentage Graph" Then
    series = "Department " & p + 1 & ":" & Departmentfield.Text & " " &
Yearfield.Text
    Departmentpercentageschart.Series.Add(series)
    'adds a cumulative series of datapoints
    Departmentpercentageschart.Series(series).Points.AddY(((fivegrade) /
dt.Rows.Count))
    Departmentpercentageschart.Series(series).Points.AddY(((fourgrade +
fivegrade) / dt.Rows.Count)))
    Departmentpercentageschart.Series(series).Points.AddY(((threegrade +
fourgrade + fivegrade) / dt.Rows.Count)))
Else
    series = "Department " & p + 1 & ":" & Departmentfield.Text & " " &
Yearfield.Text
    Departmentpercentageschart.Series.Add(series)
    'adds a non-cumulative series of datapoints
    Departmentpercentageschart.Series(series).Points.AddY(((fivegrade) /
dt.Rows.Count))
    Departmentpercentageschart.Series(series).Points.AddY(((fourgrade) /
dt.Rows.Count))
    Departmentpercentageschart.Series(series).Points.AddY(((threegrade) /
dt.Rows.Count))
End If
'sets the datapoint labels of the graph
Departmentpercentageschart.Series(series).Points(0).AxisLabel = "3"
Departmentpercentageschart.Series(series).Points(1).AxisLabel = "4"
Departmentpercentageschart.Series(series).Points(2).AxisLabel = "5"
End Sub

'this sub runs when the main menu link is clicked
Private Sub MainMenu_LinkClicked(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles MainMenu.LinkClicked
    'the following forms either close or open
    Main_Menu.Show()
    Me.Close()
End Sub
'this sub runs when the text within the level field is altered
'this sub works in a similar manner to the two subs below

```

```

Private Sub DepartmentField_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Departmentfield.SelectedIndexChanged
    'if the field is blank then the class combobox and label are not visible
    If Departmentfield.Text = "" Then
        subjectlabel.Visible = False
        SubjectField.Visible = False
    'but if the following three text fields are not blank then the
    classfieldgenerate sub is called which fills a combobox with the users selection of
    classes
    ElseIf Levelfield.Text <> "" And Yearfield.Text <> "" And Departmentfield.Text <>
    "" Then
        subjectsfieldgenerate()
    End If
End Sub
'this sub runs when the text within the level field is altered
Private Sub Levelfield_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Levelfield.SelectedIndexChanged
    If Levelfield.Text = "" Then
        subjectlabel.Visible = False
        SubjectField.Visible = False
    ElseIf Levelfield.Text <> "" And Yearfield.Text <> "" And Departmentfield.Text <>
    "" Then
        subjectsfieldgenerate()
    End If
    If Levelfield.Text = "KS2" Then
        Gradetypedefield.Enabled = False
        Gradetypedefield.Text = "Achieved"
    Else
        Gradetypedefield.Enabled = True
    End If
End Sub
'this sub runs when the text within the year field is altered
Private Sub Yearfield_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Yearfield.SelectedIndexChanged
    If Yearfield.Text = "" Then
        subjectlabel.Visible = False
        SubjectField.Visible = False
    ElseIf Levelfield.Text <> "" And Yearfield.Text <> "" And Departmentfield.Text <>
    "" Then
        subjectsfieldgenerate()
    End If
End Sub
'this sub runs when all of the fields that require the user's input have been
completed
Private Sub subjectsfieldgenerate()
    'selects the classes based on the user's choice of subject and year
    sql = "SELECT DISTINCT Department.DepartmentID, Department.Department,
    Subject.Subject FROM Department INNER JOIN Subject INNER JOIN [Level] INNER JOIN Grade ON
    [Level].LevelID = Grade.LevelID ON Subject.SubjectID = Grade.SubjectID ON
    Department.DepartmentID = Subject.DepartmentID WHERE Grade.DatasetID = '' &
    Yearfield.SelectedValue & '' and Grade.LevelID = '' & Levelfield.SelectedValue & '' and
    Department.DepartmentID = '' & Departmentfield.SelectedValue & '' ORDER BY
    Department.DepartmentID ASC"
    'stores a SQL command that uses the SQL statement to query the database that is
    found when following the connection string of con
    Using comm As SqlCommand = New SqlCommand(sql, con)
        'stores a SQL dataadareader which can read the contents of the sql query from
        the database
        Dim rs As SqlDataReader = comm.ExecuteReader
        'stores a datatable
        Dim dt As DataTable = New DataTable
        'the datatable is filled with the values that the datareader reads

```

```

        dt.Load(rs)

        'the index of each item(class) in the combobox is set as its corresponding ID
        'in the database
        SubjectField.ValueMember = "SubjectID"
        'sets the displaymember of the combobox
        SubjectField.DisplayMember = "Subject"
        'sets the datasource of the combobox
        SubjectField.DataSource = dt
    End Using
    'makes the class combobox and its label visible to the user
    subjectlabel.Visible = True
    SubjectField.Visible = True
End Sub
'this sub is run when the Clear Selection button is clicked
Private Sub Clearselection_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Clearselection.Click
    'clears all controls on this form
    Me.Controls.Clear()
    're-initializes all components of this form
    InitializeComponent()
    'then re-loads this form with all components reset
    CompareDepartment_Percentages(e, e)
End Sub
'this sub runs when the toggle labels button is clicked by the user
Private Sub labelstoggle_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles labelstoggle.Click
    'stores an index variable
    Dim s As Integer
    'if the user wishes to turn chart labels off then the following code runs
    If labelstoggle.Text = "Toggle Chart Labels Off" Then
        'this for next statemnt goes through every series that has been added by the
        user and turns off labels for that series
        For s = 0 To (p - 1)
            Departmentpercentageschart.Series(s).IsValueShownAsLabel = False
        Next
        'changes the text of the button
        labelstoggle.Text = "Toggle Chart Labels On"
        'but if the user wishes to turn chart labels on then the following code runs
    ElseIf labelstoggle.Text = "Toggle Chart Labels On" Then
        For s = 0 To (p - 1)
            Departmentpercentageschart.Series(s).IsValueShownAsLabel = True
        Next
        'changes the text of the button
        labelstoggle.Text = "Toggle Chart Labels Off"
    End If
End Sub
'this sub runs when the view number of students... button is clicked by the user
Private Sub TablesView_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles TablesView.Click
    'the following objects are set to non-visible
    Departmentpercentageschart.Visible = False
    Comparedepartmentgroup.Visible = False
    labelstoggle.Visible = False

    'the following tables and table headings become visible depending on the value of
    p as it is incremented
    Select Case (p)
        Case 1
            Table1label.Visible = True
            Seriestable1.Visible = True
        Case 2

```

```

        Table1label.Visible = True
        Seriestable1.Visible = True
        Table2label.Visible = True
        Seriestable2.Visible = True
    Case 3
        Table1label.Visible = True
        Seriestable1.Visible = True
        Table2label.Visible = True
        Seriestable2.Visible = True
        Seriestable3.Visible = True
        Table3label.Visible = True
    Case 4
        Table1label.Visible = True
        Seriestable1.Visible = True
        Table2label.Visible = True
        Seriestable2.Visible = True
        Seriestable3.Visible = True
        Table3label.Visible = True
        Table4label.Visible = True
        Seriestable4.Visible = True
    End Select
End Sub
'this sub runs when the view graph button is clicked by the user
Private Sub Graphview_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Graphview.Click
    'the following objects are set to either visible or non-visible
    'all tables and table headings are no longer visible
    Departmentpercentageschart.Visible = True
    Comparedepartmentgroup.Visible = True
    labelstoggle.Visible = True
    Table1label.Visible = False
    Table2label.Visible = False
    Table3label.Visible = False
    Table4label.Visible = False
    Seriestable1.Visible = False
    Seriestable2.Visible = False
    Seriestable3.Visible = False
    Seriestable4.Visible = False
End Sub
'this function returns a true or false state depending on whether the conditions
within this sub have been met or not
Public Function textvalidate() As Boolean
    'initially the functions state is set to true
    textvalidate = True
    'if any of the textboxes are left blank then the function returns a false state
    and the background of the textbox field changes to red, otherwise the background of the
    textbox changes to white
    If Departmentfield.Text = "" Then
        textvalidate = False
        Departmentfield.BackColor = Color.Salmon
    Else
        Departmentfield.BackColor = Color.White
    End If
    If Yearfield.Text = "" Then
        textvalidate = False
        Yearfield.BackColor = Color.Salmon
    Else
        Yearfield.BackColor = Color.White
    End If
    If Levelfield.Text = "" Then
        textvalidate = False
        Levelfield.BackColor = Color.Salmon
    End If

```

```

    Else
        Levelfield.BackColor = Color.White
    End If
End Function
'this sub deals with A2 Predictions percentages
Public Sub A2Predictions( ByVal sql2 As String)
    'stores subjects
    Dim dt As New DataTable
    'stores students
    Dim dt1 As New DataTable
    'stores studentsgrades
    Dim dt2 As New DataTable
    'stores the grid that will be displayed
    Dim dt3 As New DataTable
    'stores the subjects within that department
    Dim dt4 As New DataTable
    Dim ds As DataSet
    'stores index variables used in my for next statements
    Dim i As Integer
    Dim s As Integer
    Dim z As Integer
    'stores a variabel which counts the increments the row for dt3 datatable
    Dim rowcount As Integer = 0

    'stores the predictedvalue point score of a Grade
    Dim predictedvalue As Double
    'stores the total of all the predictedvalues
    Dim currentpredictedtotal As Double = 0
    'stores the actual predicted grade
    Dim predictedgrade As String
    'stores a subject
    Dim subject As String
    'stores a weighting variable that is applied to the predictedvalue depending on
    the subject
    Dim GCSEgradeweighting As Double
    'stores the averagepointscore for a student
    Dim averagepointscore As Double
    'stores the ID of a student
    Dim StudentID As Integer

    'queries all of the subjects and places them in alphabetical order
    sql = "SELECT Subject.Subject FROM Department INNER JOIN Subject ON
Department.DepartmentID = Subject.DepartmentID WHERE Department.DepartmentID = '" &
Departmentfield.SelectedValue & "' ORDER BY Subject ASC"
    'the dataset is set after passing the following SQL statement and table into the
    GenerateDataset sub of the public module
    ds = generateDataset(sql, "temp")
    'the dt datatable is filled with the a list of subjects from the database
    dt = ds.Tables(0)

    'the dataset is set after passing the SQL2 statement which is set depending on the
    user's filter selection, and table into the GenerateDataset sub of the public module
    ds = generateDataset(sql2, "temp")
    'the dt1 table is filled with a list of students
    dt1 = ds.Tables(0)

    'this for next loop loops through every student
    'for eachs student the dt2 datatable is filled with that students grades

    dt3.Columns.Add("Grade")

    For i = 0 To (dt1.Rows.Count - 1)

```

```

averagepointscore = 0
currentpredictedtotal = 0
StudentID = dt1.Rows(i)("StudentID").ToString()
sql = "SELECT DISTINCT Grade.GradeID, Student.Firstname, Student.Surname,
Subject.Subject, Grade.Grade, [Level].[Level], GradeType.GradeType, DatasetID.Year FROM
StudentGroup INNER JOIN Class ON StudentGroup.ClassID = Class.ClassID INNER JOIN Grade
INNER JOIN[Level] ON Grade.LevelID = [Level].LevelID INNER JOIN GradeType ON
Grade.GradeTypeID = GradeType.GradeTypeID INNER JOIN Student ON Grade.StudentID =
Student.StudentID ON StudentGroup.StudentID = Student.StudentID INNER JOIN Subject ON
Grade.SubjectID = Subject.SubjectID INNER JOIN DatasetID ON Grade.DatasetID =
DatasetID.DatasetID WHERE Student.StudentId = '" & StudentID & "' ORDER BY GradeID ASC"
ds = generateDataset(sql, "temp")
dt2 = ds.Tables(0)

'this variable is used to count the number of GCSE grades a student has
Dim count As Integer
count = 0

'this for next statement then loops through all of a students grades and then
applies a predictedvalue depending on the grade, this grade later has a weighting applied
to it
For s = 0 To (dt2.Rows.Count - 1)

    'if the value of the grade underneath the Grade header in the datatable on
    the row of the i index is an A* then this if statement runs
    If dt2.Rows(s)("Grade").ToString() = "A*" Then
        'the GCSE gradeweighting variable is received after passing the
        datatable and the row index into the GCSEGradeweighting function in the public module
        GCSEgradeweighting = GCSEGradeweighting(s, dt2)
        'the predicted value is set as a national value but then is multiplied
        by the weighting value
        predictedvalue = 58 * GCSEgradeweighting
        count = count + 1
    End If
    'all of the following if statements work like the one above for all the
    other different grades
    If dt2.Rows(s)("Grade").ToString() = "A" Then
        GCSEgradeweighting = GCSEGradeweighting(s, dt2)
        predictedvalue = 52 * GCSEgradeweighting
        count = count + 1
    End If
    If dt2.Rows(s)("Grade").ToString() = "B" Then
        GCSEgradeweighting = GCSEGradeweighting(s, dt2)
        predictedvalue = 46 * GCSEgradeweighting
        count = count + 1
    End If
    If dt2.Rows(s)("Grade").ToString() = "C" Then
        GCSEgradeweighting = GCSEGradeweighting(s, dt2)
        predictedvalue = 40 * GCSEgradeweighting
        count = count + 1
    End If
    If dt2.Rows(s)("Grade").ToString() = "D" Then
        GCSEgradeweighting = GCSEGradeweighting(s, dt2)
        predictedvalue = 34 * GCSEgradeweighting
        count = count + 1
    End If
    If dt2.Rows(s)("Grade").ToString() = "E" Then
        GCSEgradeweighting = GCSEGradeweighting(s, dt2)
        predictedvalue = 28 * GCSEgradeweighting
        count = count + 1
    End If

```

```

If dt2.Rows(s)("Grade").ToString() = "F" Then
    GCSEgradeweighting = GCSEGradeweighting(s, dt2)
    predictedvalue = 22 * GCSEgradeweighting
    count = count + 1
End If
If dt2.Rows(s)("Grade").ToString() = "G" Then
    GCSEgradeweighting = GCSEGradeweighting(s, dt2)
    predictedvalue = 16 * GCSEgradeweighting
    count = count + 1
End If
If dt2.Rows(s)("Grade").ToString() = "4" Then
    GCSEgradeweighting = GCSEGradeweighting(s, dt2)
    predictedvalue = 0
End If
If dt2.Rows(s)("Grade").ToString() = "5" Then
    GCSEgradeweighting = GCSEGradeweighting(s, dt2)
    predictedvalue = 0
End If
'the current predicted total equals the previous currentpredicted total +
the predicted value
    currentpredictedtotal = currentpredictedtotal + predictedvalue
Next

'a mean average point score for the student is created by taking the total of
all the predicted values and dividing by the number of grades
    averagepointscore = currentpredictedtotal / count
    For z = 0 To (dt.Rows.Count - 1)
        'the subject variable is set as the value of the subject field
        subject = dt.Rows(z)("Subject")
        'the actual predicted grade is set by passing the subject, row index,
        datatable and average point score to the A2predictedgrade sub in the public module
        predictedgrade = A2predictedgrade(subject, 0, dt, averagepointscore)
        'the predicted grade is then written on the row index of i underneath the
        grade header
        dt3.Rows.Add()
        dt3.Rows(rowcount)("Grade") = predictedgrade
        rowCount = rowCount + 1
    Next
Next

'this for next statement loops itself to the last row containing a grade
'it counts the number of each grade within the dataset
For i = 0 To (dt3.Rows.Count - 1)
    'if there is a 1 under any of the columns in the dataset for each row then the
    count of that corresponding grade increases by 1

    If dt3.Rows(i)("Grade").ToString() = "A*" Then
        Astargrade = Astargrade + 1
    End If
    If dt3.Rows(i)("Grade").ToString() = "A" Then
        Agrade = Agrade + 1
    End If
    If dt3.Rows(i)("Grade").ToString() = "B" Then
        Bgrade = Bgrade + 1
    End If
    If dt3.Rows(i)("Grade").ToString() = "C" Then
        Cgrade = Cgrade + 1
    End If
    If dt3.Rows(i)("Grade").ToString() = "D" Then
        Dgrade = Dgrade + 1
    End If
    If dt3.Rows(i)("Grade").ToString() = "E" Then

```

```

        Egrade = Egrade + 1
    End If
    If dt3.Rows(i)("Grade").ToString() = "F" Then
        Fgrade = Fgrade + 1
    End If
Next

'sets the value of the text labels within the table that corresponds to the index
value of p
If p = 0 Then
    'the text values of the labels in the table when the tableviewbutton is
    clicked are set to their corresponding number of grades counted from the dataset
    Utable1.Text = "U"
    tbl1g.Visible = False
    tbl1ggrade.Visible = False
    tbl1astar.Text = Astargrade & " out of " & dt3.Rows.Count
    tbl1a.Text = Agrade & " out of " & dt3.Rows.Count
    tbl1b.Text = Bgrade & " out of " & dt3.Rows.Count
    tbl1c.Text = Cgrade & " out of " & dt3.Rows.Count
    tbl1d.Text = Dgrade & " out of " & dt3.Rows.Count
    tbl1e.Text = Egrade & " out of " & dt3.Rows.Count
    tbl1f.Text = Fgrade & " out of " & dt3.Rows.Count
    'this sets the label of the table, so the user knows what the data in the
    table correlates to
    Table1label.Text = "Department " & p + 1 & ":" & Departmentfield.Text & " " &
Yearfield.Text & " A2 Predicts"
    'this sets the atoc label text of the table by adding up all the grades from
    A* to c and then dividing by the total number of grades in the dataset, which is then
    converted to a percentage and displayed as a percentage
    tbl1atoc.Text = (Agrade + Bgrade + Cgrade + Astargrade) & " out of " &
dt3.Rows.Count & " (~" & Int(((Agrade + Bgrade + Cgrade + Astargrade) / dt3.Rows.Count) *
100) & "%)"
End If
If p = 1 Then
    Utable2.Text = "U"
    tbl2g.Visible = False
    tbl2ggrade.Visible = False
    tbl2astar.Text = Astargrade & " out of " & dt3.Rows.Count
    tbl2a.Text = Agrade & " out of " & dt3.Rows.Count
    tbl2b.Text = Bgrade & " out of " & dt3.Rows.Count
    tbl2c.Text = Cgrade & " out of " & dt3.Rows.Count
    tbl2d.Text = Dgrade & " out of " & dt3.Rows.Count
    tbl2e.Text = Egrade & " out of " & dt3.Rows.Count
    tbl2f.Text = Fgrade & " out of " & dt3.Rows.Count
    Table2label.Text = "Department " & p + 1 & ":" & Departmentfield.Text & " " &
Yearfield.Text & " A2 Predicts"
    tbl2atoc.Text = (Agrade + Bgrade + Cgrade + Astargrade) & " out of " &
dt3.Rows.Count & " (~" & Int(((Agrade + Bgrade + Cgrade + Astargrade) / dt3.Rows.Count) *
100) & "%)"
End If
If p = 2 Then
    Utable3.Text = "U"
    tbl3g.Visible = False
    tbl3ggrade.Visible = False
    tbl3astar.Text = Astargrade & " out of " & dt3.Rows.Count
    tbl3a.Text = Agrade & " out of " & dt3.Rows.Count
    tbl3b.Text = Bgrade & " out of " & dt3.Rows.Count
    tbl3c.Text = Cgrade & " out of " & dt3.Rows.Count
    tbl3d.Text = Dgrade & " out of " & dt3.Rows.Count
    tbl3e.Text = Egrade & " out of " & dt3.Rows.Count
    tbl3f.Text = Fgrade & " out of " & dt3.Rows.Count

```

```

Table3label.Text = "Department " & p + 1 & ":" & Departmentfield.Text & " " &
Yearfield.Text & " A2 Predicts"
tbl3atoc.Text = (Agrade + Bgrade + Cgrade + Astargrade) & " out of " &
dt3.Rows.Count & " (~" & Int(((Agrade + Bgrade + Cgrade + Astargrade) / dt3.Rows.Count) * 100) & "%)"
End If
If p = 3 Then
    Utable4.Text = "U"
    tbl4g.Visible = False
    tbl4ggrade.Visible = False
    tbl4astar.Text = Astargrade & " out of " & dt3.Rows.Count
    tbl4a.Text = Agrade & " out of " & dt3.Rows.Count
    tbl4b.Text = Bgrade & " out of " & dt3.Rows.Count
    tbl4c.Text = Cgrade & " out of " & dt3.Rows.Count
    tbl4d.Text = Dgrade & " out of " & dt3.Rows.Count
    tbl4e.Text = Egrade & " out of " & dt3.Rows.Count
    tbl4f.Text = Fgrade & " out of " & dt3.Rows.Count
    Table4label.Text = "Department " & p + 1 & ":" & Departmentfield.Text & " " & Yearfield.Text & " A2 Predicts"
    tbl4atoc.Text = (Agrade + Bgrade + Cgrade + Astargrade) & " out of " & dt3.Rows.Count & " (~" & Int(((Agrade + Bgrade + Cgrade + Astargrade) / dt3.Rows.Count) * 100) & "%)"
End If

'this if statement selects what type of graph to display based on the user's selection
If Graphtypefield.Text = "Cumulative Grade Percentage Graph" Then
    'name of the series is set dynamically depending on users choices
    series = "Department " & p + 1 & ":" & Departmentfield.Text & " " & Yearfield.Text & " A2 Predicts"
    'adds a series to the graph
    Departmentpercentageschart.Series.Add(series)
    'adds datapoints to the series of the graph, these points work cumulatively so that each following data point adds all the previous data points values to it
    Departmentpercentageschart.Series(series).Points.AddY(Astargrade / dt3.Rows.Count)
    Departmentpercentageschart.Series(series).Points.AddY((Agrade + Astargrade) / dt3.Rows.Count)
    Departmentpercentageschart.Series(series).Points.AddY((Bgrade + Astargrade + Agrade) / dt3.Rows.Count)
    Departmentpercentageschart.Series(series).Points.AddY((Cgrade + Astargrade + Agrade + Bgrade) / dt3.Rows.Count)
    Departmentpercentageschart.Series(series).Points.AddY((Dgrade + Astargrade + Agrade + Bgrade + Cgrade) / dt3.Rows.Count)
    Departmentpercentageschart.Series(series).Points.AddY((Egrade + Dgrade + Astargrade + Agrade + Bgrade + Cgrade) / dt3.Rows.Count)
    Departmentpercentageschart.Series(series).Points.AddY((Fgrade + Egrade + Dgrade + Astargrade + Agrade + Bgrade + Cgrade) / dt3.Rows.Count)
Else
    'name of the series is set dynamically depending on users choices
    series = "Department " & p + 1 & ":" & Departmentfield.Text & " " & Yearfield.Text & " A2 Predicts"
    'adds a series to the graph
    Departmentpercentageschart.Series.Add(series)
    'adds datapoints to the series of the graph equivalent to the number of grades counted
    Departmentpercentageschart.Series(series).Points.AddY(Astargrade / dt3.Rows.Count)
    Departmentpercentageschart.Series(series).Points.AddY(Agrade / dt3.Rows.Count)
    Departmentpercentageschart.Series(series).Points.AddY(Bgrade / dt3.Rows.Count)
    Departmentpercentageschart.Series(series).Points.AddY(Cgrade / dt3.Rows.Count)
    Departmentpercentageschart.Series(series).Points.AddY(Dgrade / dt3.Rows.Count)

```

```

        Departmentpercentageschart.Series(series).Points.AddY(Egrade / dt3.Rows.Count)
        Departmentpercentageschart.Series(series).Points.AddY(Fgrade / dt3.Rows.Count)
    End If

    'sets the labels of the datapoints on the series
    Departmentpercentageschart.Series(series).Points(0).AxisLabel = "A*"
    Departmentpercentageschart.Series(series).Points(1).AxisLabel = "A"
    Departmentpercentageschart.Series(series).Points(2).AxisLabel = "B"
    Departmentpercentageschart.Series(series).Points(3).AxisLabel = "C"
    Departmentpercentageschart.Series(series).Points(4).AxisLabel = "D"
    Departmentpercentageschart.Series(series).Points(5).AxisLabel = "E"
    Departmentpercentageschart.Series(series).Points(6).AxisLabel = "U"
End Sub
'this sub runs when the view percentages button is clicked by the user
Private Sub Addsearch_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Addsearch.Click
    'if the state of the txtvalidate function is true then the following code runs
    If textvalidate() = True Then
        'this if statement validates the users selection, since only english language
        or maths are stored for KS2, the program prevents the user from displaying an empty graph
        for KS2 if any other subject is chosen. An appropriate error message is also displayed
        If Levelfield.Text = "KS2" And SubjectField.Text <> "English Language" And
SubjectField.Text <> "Maths" Then
            MsgBox("That subject is not a part of that KeyStage(Level)",
MsgBoxStyle.Information, MessageBoxButtons.OK)
        Else
            'prevents the user from adding more than 4 series
            If p = 4 Then
                MsgBox("You cannot compare any more than 4 Chart series")
            Else
                'the count of the grade variables are reset to 0
                Astargrade = 0
                Agrade = 0
                Bgrade = 0
                Cgrade = 0
                Dgrade = 0
                Egrade = 0
                Fgrade = 0
                Ggrade = 0
                threegrade = 0
                fourgrade = 0
                fivegrade = 0

                'stores a dataset
                Dim ds As New DataSet
                'stores a datatable
                Dim dt As New DataTable

                'the current graph series and title labels are cleared if they have
already been set
                Departmentpercentageschart.Titles.Clear()
                'the toggle labels button becomes visible
                labelstoggle.Visible = True

                'calls the Loadgraphic class
                Dim objPlsWait As New clsOPAWaitScreen
                objPlsWait.ShowWaitScreen("")

                If Gradetypefield.Text = "A2 Predicted" Then
                    tablettitle1.Text = "Number of Student's Predicted Grade"
                    tablettitle2.Text = "Number of Student's Predicted Grade"

```

```

        tablettitle3.Text = "Number of Student's Predicted Grade"
        tablettitle4.Text = "Number of Student's Predicted Grade"

        'sets the chart label text depending on the users graphtype
selection
        If Graphtypefield.Text = "Cumulative Grade Percentage Graph" Then
            chartlabel.Text = "Cumulative Predictions by Grade (%)"
        Else
            chartlabel.Text = "Percentage of Students Predicted a Grade
(%)"
        End If

        'selects all students within the year the user has selected in the
yearfield
        Dim sql2 As String
        sql2 = "SELECT DISTINCT Student.StudentID, Student.Firstname,
Student.Surname FROM StudentGroup INNER JOIN Class ON StudentGroup.ClassID = Class.ClassID
INNER JOIN Grade INNER JOIN[Level] ON Grade.LevelID = [Level].LevelID INNER JOIN GradeType
ON Grade.GradeTypeID = GradeType.GradeTypeID INNER JOIN Student ON Grade.StudentID =
Student.StudentID ON StudentGroup.StudentID = Student.StudentID INNER JOIN Subject ON
Grade.SubjectID = Subject.SubjectID INNER JOIN DatasetID ON Grade.DatasetID =
DatasetID.DatasetID WHERE Student.DatasetID = '" & Yearfield.SelectedValue & "' ORDER BY
Student.StudentID ASC"
        'passes the SQL statement to the A2predictions sub
A2Predictions(sql2)

        Else
            'sets the chart label text depending on the users graphtype
selection
            If Graphtypefield.Text = "Cumulative Grade Percentage Graph" Then
                chartlabel.Text = "Cumulative Percentage by Grade (%)"
            Else
                chartlabel.Text = "Percentage of Students Achieving a Grade
(%)"
            End If

            If Levelfield.Text = "KS4" Then
                'if the levelfield value is KS4 then this code runs
                'runs the KS4 percentages sub
                KS4percentages()
                'the graph becomes visible
                Departmentpercentageschart.Visible = True
            Else
                'runs the KS2 percentages sub
                KS2percentages()
                'the graph becomes visible
                Departmentpercentageschart.Visible = True
            End If
        End If

        'disables the legend from the graph
        Departmentpercentageschart.Series(p).IsVisibleInLegend = True
        'shows the values of datapoints
        Departmentpercentageschart.Series(p).IsValueShownAsLabel = True
        'sets the colour of the datapoint labels as maroon
        Departmentpercentageschart.Series(p).ChartType =
DataVisualization.Charting.SeriesChartType.Line
        'changes the chart colour to skyblue
        Departmentpercentageschart.Series(p).LabelForeColor = Color.Maroon
        'changes the chart to a 3D bar style
        Departmentpercentageschart.Series(0).Color = Color.DarkGreen

```

```

'sets the minimum and maximum axis
Departmentpercentageschart.ChartAreas(0).AxisY.Minimum = 0
Departmentpercentageschart.ChartAreas(0).AxisY.Maximum = 1
'sets the format of the chart labels
Departmentpercentageschart.ChartAreas(0).AxisY.LabelStyle.Format =
"#%"
'sets the format of the series labels
Departmentpercentageschart.Series(p).LabelFormat = "##.##" & "%"
'sets the colour of the axis labels
Departmentpercentageschart.ChartAreas(0).AxisX.LabelStyle.ForeColor =
Color.Maroon
Departmentpercentageschart.ChartAreas(0).AxisY.LabelStyle.ForeColor =
Color.Maroon
'sets the background of the chart as transparent
Departmentpercentageschart.ChartAreas(0).BackColor = Color.Transparent
'increments the value of p by 1 as a new series is added by the user
p = p + 1
'graph becomes visible
Departmentpercentageschart.Visible = True
'if the user selects KS4 then the groupview groupbox is visible
If Levelfield.Text = "KS4" Then
    Groupview.Visible = True
End If

'these fields are no longer enables so that the user cannot add a
series for KS2 and KS4, or add a series of one graph type and a series for the other
graph type
Gradetypefield.Enabled = False
Graphtypefield.Enabled = False
Levelfield.Enabled = False

'closes the loadgraphic wait and resets its objplswait variable
objPlsWait.CloseWaitScreen()
objPlsWait = Nothing
Addsearch.Text = "Add Series"
End If
End If
Else
    'if the state of the txtvalidate function is not true then the following error
message displays and nothing else happens
    MsgBox("One or more textboxes have not been completed",
MsgBoxStyle.Information, MessageBoxButtons.OK)
End If
End Sub

'deals with the print form button
Private Sub printbutton_Click(sender As System.Object, e As System.EventArgs) Handles
printbutton.Click
    'sets the printer settings to default
    PrintForm.PrinterSettings = PrintDialog.PrinterSettings
    If PrintDialog.ShowDialog() = DialogResult.OK Then
        'allows the user to change print settings
        PrintForm.PrinterSettings = PrintDialog.PrinterSettings
        'changes the orientation of the printed document to landscape
        Me.PrintForm.PrinterSettings.DefaultPageSettings.Landscape = True
        'refreshes the form so that the printdialog is not printed along with the form
        Me.Refresh()
        'prints the current form
        Me.PrintForm.Print()
        'Percentageschart.Printing.Print(True)
    End If
End Sub

```

[End Class](#)

My Classes

The my classes form is only accessible by users who are non-admins. This form is responsible for displaying all of the classes of the teacher that has accessed the system. So when this form is displayed all of the teacher's classes are displayed. From here a teacher can view the students within their class and then choose whether to remove that student from their class. They can also add a new student to their class in which case a new form will display allowing the teacher to select a student and add that student to one of their classes. Furthermore, a teacher can edit the details of one of their classes from this form. A printed hard copy of this form can also be produced for a physical copy of records within the data grid.

The screenshot shows a Windows application window titled "My classes". The menu bar includes "Main Menu" and "My Classes". The toolbar contains a "Print" icon. The main area is titled "My Classes" and features a "ClearSelection" button. Below is a data grid showing three class entries:

ClassGroup	Subject	Department	Year	Firstname	Surname	View Current Students	Add a Student to this Class
Class Group 98	English Language	English	2014	Andrea	Carr	View Students	Add Student
Class Group 101	English Literature	English	2014	Andrea	Carr	View Students	Add Student
Class Group 139	Biology	Science	2014	Andrea	Carr	View Students	Add Student

```

banner System.Windows.Forms.PictureBox
Classeslink System.Windows.Forms.LinkLabel
classgrid System.Windows.Forms.DataGridView
Clearselection System.Windows.Forms.Button
Formheader System.Windows.Forms.Label
linkarrow System.Windows.Forms.Label
MainMenu System.Windows.Forms.LinkLabel
Myclasses System.Windows.Forms.Form
printbutton System.Windows.Forms.Button
PrintDialog System.Windows.Forms.PrintDialog
PrintForm Microsoft.VisualBasic.PowerPacks.Printing.PrintForm
Viewstudentsgrid System.Windows.Forms.DataGridView
  
```

Parameter Name	Description
MyClasses	This sub handles the loading of the form. It connects to the database and displays the classes of the teacher that has logged in, in the data grid.
MainMenu_LinkClicked	When this link is clicked the main menu is displayed.
printbutton_Click	This sub allows the user to print a hard copy of the form when the print icon is clicked, additional print dialog options are then displayed allowing the user to choose printer and number of copies amongst other settings.
Clearselection_Click	Creates a new instance of every object on the form (essentially re-loads the form).
classgrid_CellContentClick	This sub handles the instance one of the buttons on the class grid is clicked by the user, depending on the column index of the button different actions take place. For example if the edit class button is clicked then the edit class form will appear with the details of the class the user wants to edit.
delete_CellContentClick	This sub handles the instance when one of the remove class buttons that is a part of the viewstudentsgrid is clicked. It displays the remove student form with the details of the student the user wants to remove passed into the remove student form.

```

Public Class Myclasses
    Private Sub MyClasses(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Me.Load
        classgrid.Columns.Clear()
        DatabaseConnect()
        'stores a dataset variable
        Dim ds As DataSet
        'sets the SQL statement
        sql = "SELECT DISTINCT Class.ClassID, Class.ClassGroup, Subject.Subject,
Department.Department, DatasetID.Year, Teacher.Firstname, Teacher.Surname FROM Department
INNER JOIN Subject ON Department.DepartmentID = Subject.DepartmentID INNER JOIN Class
INNER JOIN DatasetID ON Class.DatasetID = DatasetID.DatasetID ON Subject.SubjectID =
Class.SubjectID INNER JOIN Teacher INNER JOIN Teaches ON Teacher.TeacherID =
Teaches.TeacherID ON Class.TeacherID = Teacher.TeacherID WHERE Teacher.LoginID = '" &
moduleuserid & "' and DatasetID.DatasetID = '" & moduleDataset & "' ORDER BY
Class.ClassID ASC"
        'sets the datagrid as visible
        classgrid.Visible = True
        'the dataset is set after passing the following SQL statement and table into the
GenerateDataset sub of the public module
        ds = generateDataset(sql, "Class")
        'sets the datasource of the datagrid
        classgrid.DataSource = ds.Tables(0)
        'sets the datamember of the datagrid
        classgrid.DataMember = ""
        'hides the ClassID column from the user
        classgrid.Columns("ClassID").Visible = False

        'creates a new variable that stores information for a button column that can be
added to the datagrid
        Dim StudentColumn As New DataGridViewButtonColumn
        With StudentColumn
            'sets the header text of the button column
            .HeaderText = "View Current Students"
            'sets the name of the button column
            .Name = "Details"
            'sets the text displayed on the buttons of the button column in the datagrid
            .Text = "View Students"
            .UseColumnTextForButtonValue = True
        End With

        Dim studentadd As New DataGridViewButtonColumn
        With studentadd
            .HeaderText = "Add a Student to this Class"
            .Name = "Details"
            .Text = "Add Student"
            .UseColumnTextForButtonValue = True
        End With

        'adds the button columns to the datagrid
        classgrid.Columns.Add(StudentColumn)
        classgrid.Columns.Add(studentadd)

        'auto resizes the column widths of the datagrid
        classgrid.AutoResizeColumns()
    End Sub
    'shows the main menu and closes the my classes form
    Private Sub MainMenu_LinkClicked(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles MainMenu.LinkClicked
        Main_Menu.Show()
        Me.Close()
    End Sub

```

```

'deals with the print form button
Private Sub printbutton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles printbutton.Click
    'sets the printer settings to default
    PrintForm.PrinterSettings = PrintDialog.PrinterSettings
    If PrintDialog.ShowDialog() = DialogResult.OK Then
        'allows the user to change print settings
        PrintForm.PrinterSettings = PrintDialog.PrinterSettings
        'changes the orientation of the printed document to landscape
        Me.PrintForm.PrinterSettings.DefaultPageSettings.Landscape = True
        'refreshes the form so that the printdialog is not printed along with the form
        Me.Refresh()
        'prints the current form
        Me.PrintForm.Print()
    End If
End Sub
'this sub is run when the Clear Selection button is clicked
Private Sub Clearselection_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Clearselection.Click
    'clears all controls on this form
    Me.Controls.Clear()
    're-initializes all components of this form
    InitializeComponent()
    'then re-loads this form with all components reset
    MyClasses(e, e)
End Sub
'handles the instance the classesgrid buttons are clicked
Private Sub classgrid_CellContentClick(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.DataGridViewCellEventArgs) Handles classgrid.CellContentClick
    'stores ds as a dataset
    Dim ds As New DataSet
    'stores the classID
    Dim ClassID As String

    'if the selected index of the button that is clicked is the seventh column then
    'the following action is performed
    If e.ColumnIndex = 7 Then
        ClassID = classgrid.Rows(e.RowIndex).Cells("ClassID").Value
        'clears the current datagrid column selection
        classgrid.Columns.Clear()

        'sets the SQL statement query
        sql = "SELECT Class.ClassID, Student.Firstname, Student.Surname, Student.Form,
Class.ClassGroup, Subject.Subject, Department.Department, DatasetID.Year FROM StudentGroup
INNER JOIN Class ON StudentGroup.ClassID = Class.ClassID INNER JOIN Subject ON
Class.SubjectID = Subject.SubjectID INNER JOIN Department ON Subject.DepartmentID =
Department.DepartmentID INNER JOIN Student ON StudentGroup.StudentID = Student.StudentID
INNER JOIN DatasetID ON Class.DatasetID = DatasetID.DatasetID WHERE Class.ClassID = '" &
ClassID & "' ORDER BY Student.Firstname ASC, Student.Surname ASC"

        'the dataset is set after passing the following SQL statement and table into
        'the GenerateDataset sub of the public module
        ds = generateDataset(sql, "temp")
        'sets the datasource of the datagrid
        Viewstudentsgrid.DataSource = ds
        'sets the following datagrids as either visible or not visible
        classgrid.Visible = False
        Viewstudentsgrid.Visible = True
        'sets the datamember of the datagrid
        Viewstudentsgrid.DataMember = "temp"
        'makes the ClassID column not visible to the user
        Viewstudentsgrid.Columns("ClassID").Visible = False
    End If
End Sub

```

```

'creates a new variable that stores information for a button column that can
be added to the datagrid
Dim deleteStudentColumn As New DataGridViewButtonColumn
With deleteStudentColumn
    'sets the header text of the button column
    .HeaderText = "Remove this Student from this class"
    'sets the name of the button column
    .Name = "Details"
    'sets the text displayed on each button of the button column
    .Text = "Remove Student"
    .UseColumnTextForButtonValue = True
End With

'adds the button column to the datagrid
Viewstudentsgrid.Columns.Add(deleteStudentColumn)
'automatically resizes all column widths in the datagrid
Viewstudentsgrid.AutoResizeColumns()
End If

'if the selected index of the button that is clicked is the eighth column then the
following action is performed
If e.ColumnIndex = 8 Then
    Add_Student_Class.Close()
    'returns the value of the ClassID that is on the same row as the button
    clicked and is under the column with the header of classID
    ClassID = classgrid.Rows(e.RowIndex).Cells("ClassID").Value

    Dim AddNewStudenttoClass As New Edit_Classes
    'sets the ClassID of the add student Class form as the ClassID of this form
    Add_Student_Class.ClassID = ClassID
    Add_Student_Class.Show()
End If

End Sub
'when the remove student from class button is clicked, the code within the following
sub is run
Private Sub delete_CellContentClick(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.DataGridViewCellEventArgs) Handles Viewstudentsgrid.CellContentClick
    'stores ds as a new dataset
    Dim ds As New DataSet
    'stores the ClassID
    Dim ClassID As String
    'stores the firstname of a student
    Dim fname As String
    'Stores the surname of the student
    Dim sname As String
    'stores the form of the student
    Dim form As String

    If e.ColumnIndex = 8 Then
        Remove_Student_Class.Close()
        'sets the value of these four variables as the value of the cell with the
        selected column name that is on the same row as the button that has been clicked
        ClassID = Viewstudentsgrid.Rows(e.RowIndex).Cells("ClassID").Value
        fname = Viewstudentsgrid.Rows(e.RowIndex).Cells("Firstname").Value
        sname = Viewstudentsgrid.Rows(e.RowIndex).Cells("Surname").Value
        form = Viewstudentsgrid.Rows(e.RowIndex).Cells("form").Value

        Dim removestudentfromclass As New Remove_Student_Class
        'sets the following four variables of the remove student class form as the
        values of their variables found from thedatagrid of this form

```

```
Remove_Student_Class.ClassID = ClassID
Remove_Student_Class.Fname = fname
Remove_Student_Class.Sname = sname
Remove_Student_Class.Form = form
Remove_Student_Class.Show()
End If
End Sub
End Class
```

MyClassGrades

The my class Gradesform is only accessible by users who are non-admins. This form is responsible for displaying all of the classes of the teacher that has accessed the system. So when this form is displayed all of the teacher's classes are displayed. From the data grid a teacher can choose to view all the grades within one of their classes. When these grades become visible in the data grid, a user can then choose to edit any one of the grades from one of their classes. This allows a teacher to update the grade of a student within one of their classes. A printed hard copy of this form can also be produced for a physical copy of records within the data grid.

Firstname	Surname	Subject	Grade	Level	GradeType	Year	Edit Class Grade	View all Student's Grades
Holly	LANSBURY	English Literature	B	KS4	Achieved	2015	Edit Grade	View Grades
Jack	KENWORTHY	English Literature	A	KS4	Achieved	2015	Edit Grade	View Grades
Jack	KING	English Literature	A	KS4	Achieved	2015	Edit Grade	View Grades
Jack	KINGSCOTT	English Literature	B	KS4	Achieved	2015	Edit Grade	View Grades
Jack	LAMB	English Literature	A	KS4	Achieved	2015	Edit Grade	View Grades
Jacob	JONES	English Literature	A	KS4	Achieved	2015	Edit Grade	View Grades
Jacob	JOPP	English Literature	A	KS4	Achieved	2015	Edit Grade	View Grades
James	HUNG	English Literature	B	KS4	Achieved	2015	Edit Grade	View Grades
James	JOHNSON	English Literature	C	KS4	Achieved	2015	Edit Grade	View Grades
James	JOHNSON	English Literature	A*	KS4	Achieved	2015	Edit Grade	View Grades
Jamie	HUGHES	English Literature	B	KS4	Achieved	2015	Edit Grade	View Grades
Jasmine	HOWE	English Literature	C	KS4	Achieved	2015	Edit Grade	View Grades
Jason	HOWARD	English Literature	A	KS4	Achieved	2015	Edit Grade	View Grades
Jed	HORTON	English Literature	A	KS4	Achieved	2015	Edit Grade	View Grades
Jennifer	HOLMES	English Literature	A	KS4	Achieved	2015	Edit Grade	View Grades
Jennifer	HOME	English Literature	B	KS4	Achieved	2015	Edit Grade	View Grades
Jennifer	HORNER	English Literature	A*	KS4	Achieved	2015	Edit Grade	View Grades

```

banner System.Windows.Forms.PictureBox
Classeslink System.Windows.Forms.LinkLabel
Classgrid System.Windows.Forms.DataGridView
Clearselection System.Windows.Forms.Button
Formheader System.Windows.Forms.Label
linkarrow System.Windows.Forms.Label
MainMenu System.Windows.Forms.LinkLabel
MyClassGrades System.Windows.Forms.Form
printbutton System.Windows.Forms.Button
PrintDialog System.Windows.Forms.PrintDialog
PrintForm Microsoft.VisualBasic.PowerPacks.Printing.PrintForm

```

Parameter Name	Description
MyClassGrades	This sub handles the loading of the form. It connects to the database and displays the classes of the teacher that has logged in, in the data grid.
printbutton_Click	This sub allows the user to print a hard copy of the form when the print icon is clicked, additional print dialog options are then displayed allowing the user to choose printer and number of copies amongst other settings.
Clearselection_Click	Creates a new instance of every object on the form (essentially re-loads the form).
Classgrid_CellContentClick	This sub handles the instance one of the buttons on the class grid is clicked by the user, depending on the column index of the button different actions take place.
MainMenu_LinkClicked	When this link is clicked the main menu is displayed.

```

Public Class MyClassGrades
    Private Sub MyClassGrades(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Me.Load
        Classgrid.Columns.Clear()
        DatabaseConnect()
        'stores a dataset variable
        Dim ds As DataSet
        'sets the SQL statement
        sql = "SELECT DISTINCT Class.ClassID, Class.ClassGroup, Subject.Subject,
Department.Department, DatasetID.Year, Teacher.Firstname, Teacher.Surname FROM Department
INNER JOIN Subject ON Department.DepartmentID = Subject.DepartmentID INNER JOIN Class
INNER JOIN DatasetID ON Class.DatasetID = DatasetID.DatasetID ON Subject.SubjectID =
Class.SubjectID INNER JOIN Teacher INNER JOIN Teaches ON Teacher.TeacherID =
Teaches.TeacherID ON Class.TeacherID = Teacher.TeacherID WHERE Teacher.LoginID = '" &
moduleuserid & "' and DatasetID.DatasetID = '" & moduleDataset & "' ORDER BY
Class.ClassID ASC"
        'sets the datagrid as visible
        Classgrid.Visible = True
        'the dataset is set after passing the following SQL statement and table into the
GenerateDataset sub of the public module
        ds = generateDataset(sql, "Class")
        'sets the datasource of the datagrid
        Classgrid.DataSource = ds.Tables(0)
        'sets the datamember of the datagrid
        Classgrid.DataMember = ""
        'hides the ClassID column from the user
        Classgrid.Columns("ClassID").Visible = False

        'creates a new variable that stores information for a button column that can be
added to the datagrid
        Dim viewgrades As New DataGridViewButtonColumn
        With viewgrades
            'sets the header text of the button column
            .HeaderText = "View Class Grades"
            'sets the name of the button column
            .Name = "Details"
            'sets the text displayed on the buttons of the button column in the datagrid
            .Text = "View Grades"
            .UseColumnTextForButtonValue = True
        End With
        'adds the button columns to the datagrid
        Classgrid.Columns.Add(viewgrades)
        'auto resizes the column widths of the datagrid
        Classgrid.AutoResizeColumns()
    End Sub
    'deals with the print form button
    Private Sub printbutton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles printbutton.Click
        'sets the printer settings to default
        PrintForm.PrinterSettings = PrintDialog.PrinterSettings
        If PrintDialog.ShowDialog() = DialogResult.OK Then
            'allows the user to change print settings
            PrintForm.PrinterSettings = PrintDialog.PrinterSettings
            'changes the orientation of the printed document to landscape
            Me.PrintForm.PrinterSettings.DefaultPageSettings.Landscape = True
            'refreshes the form so that the printdialog is not printed along with the form
            Me.Refresh()
            'prints the current form
            Me.PrintForm.Print()
        End If
    End Sub

```

```

'this sub is run when the Clear Selection button is clicked
Private Sub Clearselection_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Clearselection.Click
    'clears all controls on this form
    Me.Controls.Clear()
    're-initializes all components of this form
    InitializeComponent()
    'then re-loads this form with all components reset
    MyClassGrades(e, e)
End Sub

Private Sub Classgrid_CellContentClick(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.DataGridViewCellEventArgs) Handles Classgrid.CellContentClick
    'stores ds as a dataset
    Dim ds As New DataSet
    'stores the classID
    Dim ClassID As String
    'stores a GradeID
    Dim GradeID As String
    'stores the subject string
    Dim subject As String
    'stores a studentID
    Dim studentid As Integer

    'if the selected index of the button that is clicked is the seventh column then
    'the following action is performed
    If e.ColumnIndex = 7 Then
        ClassID = Classgrid.Rows(e.RowIndex).Cells("ClassID").Value
        subject = Classgrid.Rows(e.RowIndex).Cells("Subject").Value
        'clears the current datagrid column selection
        Classgrid.Columns.Clear()

        'sets the SQL statement query
        sql = "SELECT DISTINCT Grade.GradeID, Student.StudentID, Student.Firstname,
        Student.Surname, Subject.Subject, Grade.Grade, [Level].[Level], GradeType.GradeType,
        DatasetID.Year FROM Department INNER JOIN Subject ON Department.DepartmentID =
        Subject.DepartmentID CROSS JOIN Student LEFT OUTER JOIN StudentGroup INNER JOIN Class ON
        StudentGroup.ClassID = Class.ClassID INNER JOIN Grade ON Class.SubjectID = Grade.SubjectID
        INNER JOIN DatasetID ON Class.DatasetID = DatasetID.DatasetID INNER JOIN [Level] ON
        Grade.LevelID = [Level].LevelID INNER JOIN GradeType ON Grade.GradeTypeID =
        GradeType.GradeTypeID ON Student.StudentID = Grade.StudentID AND Student.StudentID =
        StudentGroup.StudentID WHERE Class.ClassID = '" & ClassID & "' AND level.Level = 'KS4' AND
        Subject.Subject = '" & subject & "' ORDER BY Student.Firstname ASC, Student.Surname ASC"
        'the dataset is set after passing the following SQL statement and table into
        'the GenerateDataset sub of the public module
        ds = generateDataset(sql, "temp")
        'sets the datasource of the datagrid
        Classgrid.DataSource = ds
        'sets the following datagrids as either visible or not visible
        Classgrid.Visible = True
        'sets the datamember of the datagrid
        Classgrid.DataMember = "temp"

        Classgrid.Columns("GradeID").Visible = False
        Classgrid.Columns("StudentID").Visible = False

        'creates a new variable that stores information for a button column that can
        'be added to the datagrid
        Dim editgrade As New DataGridViewButtonColumn
        With editgrade
            'sets the header text of the button column
            .HeaderText = "Edit Class Grade"
        End With
    End If
End Sub

```

```

'sets the name of the button column
.Name = "Details"
'sets the text displayed on the buttons of the button column in the
datagrid
.Text = "Edit Grade"
.UseColumnTextForButtonValue = True
End With

'creates a new variable that stores information for a button column that can
be added to the datagrid
Dim allgrades As New DataGridViewButtonColumn
With allgrades
    'sets the header text of the button column
    .HeaderText = "View all Student's Grades"
    'sets the name of the button column
    .Name = "Details"
    'sets the text displayed on the buttons of the button column in the
datagrid
    .Text = "View Grades"
    .UseColumnTextForButtonValue = True
End With

'adds the button column to the datagrid
Classgrid.Columns.Add(editgrade)
Classgrid.Columns.Add(allgrades)

'auto resizes the column selection
Classgrid.AutoResizeColumns()
End If

If e.ColumnIndex = 9 Then
    Edit_Grade.Close()
    'the GradeID variable is set as teh value of the cell that is on the same row
    'as the button that is clicked and is under the GradeID header column
    GradeID = Classgrid.Rows(e.RowIndex).Cells("GradeID").Value

    'the GradeID of the Edit Grade Form is set to the Grade ID that is found from
    this form
    Edit_Grade.GradeID = GradeID
    'The edit grade form is displayed
    Edit_Grade.Show()
End If

If e.ColumnIndex = 10 Then
    'the studentID variable is set as the STudentID that is on the same row as the
    button clicked by the user
    studentid = Classgrid.Rows(e.RowIndex).Cells("StudentID").Value
    'clears the current selection of columns in the datagrid
    Classgrid.Columns.Clear()

    'works like the example above in the addsearch sub and displays the results of
    the SQL statement query below in addgradegrid which is then set as visible
    sql = "SELECT DISTINCT Grade.GradeID, Student.Firstname, Student.Surname,
Subject.Subject, Grade.Grade, [Level].[Level], GradeType.GradeType, DatasetID.Year FROM
StudentGroup INNER JOIN Class ON StudentGroup.ClassID = Class.ClassID INNER JOIN Grade
INNER JOIN[Level] ON Grade.LevelID = [Level].LevelID INNER JOIN GradeType ON
Grade.GradeTypeID = GradeType.GradeTypeID INNER JOIN Student ON Grade.StudentID =
Student.StudentID ON StudentGroup.StudentID = Student.StudentID INNER JOIN Subject ON
Grade.SubjectID = Subject.SubjectID INNER JOIN DatasetID ON Grade.DatasetID =
DatasetID.DatasetID WHERE Student.StudentId = '" & studentid & "' ORDER BY GradeID ASC"
    ds = generateDataset(sql, "temp")
    Classgrid.DataSource = ds

```

```
Classgrid.Visible = True
Classgrid.DataMember = "temp"
Classgrid.Columns("GradeID").Visible = False
'auto resizes the column selection
Classgrid.AutoResizeColumns()
End If

End Sub
'shows the main menu and closes the my classes form
Private Sub MainMenu_LinkClicked(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles MainMenu.LinkClicked
    Main_Menu.Show()
    Me.Close()
End Sub
End Class
```

Module1

I have used a module within my system so that I could break down a problem into smaller tasks. These tasks I could then break down into subtasks. This divide and conquer strategy helped me to make daunting programming tasks much more feasible. An advantage of using modular code was so that I could re-use commonly used functions and processes within my system allowing for much better code-re-usability. It also makes managing the system much easier for future developers as there is only one instance of commonly used code.

```
'Imports the following system code commands so that I can use them appropriately
Imports System.Data.SqlClient
Imports System.Data
Imports System.Configuration
Imports System.Data.DataTable
Imports System
Imports System.IO
Imports System.Text
'since this is a module any subs within this module can be called from any forms
Public Module Module1
    'stores the connection string as a public variable that can be used across all forms
    Public connectionString As String
    'stores the SQL connection as a public variable that can be used across all forms
    Public con As SqlConnection
    'Stores a SQL query as a public string variable that can be used across all forms
    Public sql As String
    'stores an autosize columns variable that can be used across all forms
    Public Property AutoSizeColumnsMode As DataGridViewAutoSizeColumnsMode
    'these usernames and passwords are stored so that their stored variable are not reset
    'when switching forms
    Public modulepassword As String
    Public moduleusername As String
    Public moduleuserid As Integer
    'stores the current dataset in use by the system
    Public moduleDataset As String = "0"
    Public addnewdataset As Boolean
    Sub DatabaseConnect()
        'stores the connection path
        Dim connectionstring As String
        'stores the SQL server connection
        Dim connection As New SqlConnection
        'sets the connection path
        connectionstring = "Data Source=<>;Initial Catalog=<>;User ID=<>;Password=<>"'
        'the connection is then set as a SQL connection that uses the connection path of
        'the connectionstring variable
        con = New SqlConnection(connectionstring)

        'a try catch is used to prevent the program from crashing
        Try
            'attempts to connect to the database using the connection path
            con.Open()
        Catch ex As Exception
            'if the database can not be connected to then the following error message is
            displayed
            MsgBox("Can not open SQL Server connection! Make Sure Server Device is
Switched on. ")
        End Try
    End Sub
```

```

'The following function generates a combo box by using a table and field that is
passed into the function and a true/false statement to declare whether the top row of the
combobox is empty
Function GenerateCombo(table As String, field As String, needEmpty As Boolean)
    'selects distinct field from a table and then order in ascending order
    sql = "SELECT DISTINCT " & field & " FROM " & table & " ORDER BY " & field &
ASC"
    'stores a SQL datadapter variable that connects to the connection and uses the
above SQL statement to query the SQL database
    Dim dataadapter As New SqlDataAdapter(sql, con)
    'stores a dataset
    Dim FilterDS As New DataSet
    'stores a datatable
    Dim FilterDT As DataTable

    'the dataadapter fills the dataset with the results of the SQL query
    dataadapter.Fill(FilterDS, table)
    'the datatable is set as the first dataset table
    FilterDT = FilterDS.Tables(0)
    'if needempty is true then the datatable inserts an empty row at row position zero
in the combobox
    If needEmpty Then
        Dim drNewRow As DataRow = FilterDT.NewRow
        FilterDT.Rows.InsertAt(drNewRow, 0)
        FilterDT.AcceptChanges()
    End If
    'returns the datatable
    Return FilterDT
End Function

'the following function generates a dataset by passing a sql string and a table into
the function when it is called
Function generateDataset(ByVal sql As String, ByVal table As String)
    'stores a SQL datadapter variable that connects to the connection and uses the
above SQL statement to query the SQL database
    Dim dataadapter As New SqlDataAdapter(sql, con)
    'stores a dataset variable
    Dim ds As New DataSet()
    'The dataadapter fills the dataset with the result of the SQL query
    dataadapter.Fill(ds, table)
    'returns the dataset
    Return ds
End Function

'This function generates a dataset for editing class values by passing a classID into
the function
Public Function Editclassvalues(ByVal ClassID As String)
    'sets the SQL statement
    sql = "SELECT DISTINCT Class.ClassID, Teacher.Firstname, Teacher.Surname,
Class.ClassGroup, Subject.Subject, Department.Department, DatasetID.Year FROM Department
INNER JOIN Subject ON Department.DepartmentID = Subject.DepartmentID INNER JOIN Class
INNER JOIN DatasetID ON Class.DatasetID = DatasetID.DatasetID ON Subject.SubjectID =
Class.SubjectID INNER JOIN Teacher INNER JOIN Teaches ON Teacher.TeacherID =
Teaches.TeacherID ON Class.TeacherID = Teacher.TeacherID WHERE Class.ClassID = '" &
ClassID & "'"
    'stores a SQL datadapter variable that connects to the connection and uses the
above SQL statement to query the SQL database
    Dim dataadapter As New SqlDataAdapter(sql, con)
    'stores a dataset variable
    Dim ds As New DataSet
    'The dataadapter fills the dataset with the result of the SQL query
    dataadapter.Fill(ds, "temp")
    'returns the dataset
    Return ds

```

```

End Function
'This function generates a dataset for editing grade values by passing a GradeID into
the function
Public Function Editgradevalues(ByVal GradeID As String)
    'sets the SQL statement
    sql = "SELECT DISTINCT Grade.GradeID, Student.Firstname, Student.Surname,
Subject.Subject, Grade.Grade, [Level].[Level], GradeType.GradeType, DatasetID.Year FROM
StudentGroup INNER JOIN Class ON StudentGroup.ClassID = Class.ClassID INNER JOIN Grade
INNER JOIN[Level] ON Grade.LevelID = [Level].LevelID INNER JOIN GradeType ON
Grade.GradeTypeID = GradeType.GradeTypeID INNER JOIN Student ON Grade.StudentID =
Student.StudentID ON StudentGroup.StudentID = Student.StudentID INNER JOIN Subject ON
Grade.SubjectID = Subject.SubjectID INNER JOIN DatasetID ON Grade.DatasetID =
DatasetID.DatasetID WHERE Grade.GradeID = '" & GradeID & "'"
    'stores a SQL datadapter variable that connects to the connection and uses the
above SQL statement to query the SQL database
    Dim dataadapter As New SqlDataAdapter(sql, con)
    'stores a dataset variable
    Dim ds As New DataSet
    'The dataadapter fills the dataset with the result of the SQL query
    dataadapter.Fill(ds, "temp")
    'returns the dataset
    Return ds
End Function
'This function generates a dataset for adding grade values by passing a StudentID into
the function
Public Function Addgradevalues(ByVal StudentID As String)
    'sets the SQL statement
    sql = "SELECT DISTINCT Grade.GradeID, Student.Firstname, Student.Surname,
Subject.Subject, Grade.Grade, [Level].[Level], GradeType.GradeType, DatasetID.Year FROM
StudentGroup INNER JOIN Class ON StudentGroup.ClassID = Class.ClassID INNER JOIN Grade
INNER JOIN[Level] ON Grade.LevelID = [Level].LevelID INNER JOIN GradeType ON
Grade.GradeTypeID = GradeType.GradeTypeID INNER JOIN Student ON Grade.StudentID =
Student.StudentID ON StudentGroup.StudentID = Student.StudentID INNER JOIN Subject ON
Grade.SubjectID = Subject.SubjectID INNER JOIN DatasetID ON Grade.DatasetID =
DatasetID.DatasetID WHERE Student.StudentID = '" & StudentID & "'"
    'stores a SQL datadapter variable that connects to the connection and uses the
above SQL statement to query the SQL database
    Dim dataadapter As New SqlDataAdapter(sql, con)
    'stores a dataset variable
    Dim ds As New DataSet
    'The dataadapter fills the dataset with the result of the SQL query
    dataadapter.Fill(ds, "temp")
    'returns the dataset
    Return ds
End Function
'This function generates a dataset for editing student values by passing a StudentID
into the function
Public Function StudentEditValues(ByVal StudentID As String)
    'sets the SQL statement
    sql = "Select * FROM Student WHERE StudentID = " & StudentID & ";"
    'stores a SQL datadapter variable that connects to the connection and uses the
above SQL statement to query the SQL database
    Dim dataadapter As New SqlDataAdapter(sql, con)
    'stores a dataset variable
    Dim ds As New DataSet
    'The dataadapter fills the dataset with the result of the SQL query
    dataadapter.Fill(ds, "Student")
    'returns the dataset
    Return ds
End Function
'This function generates a dataset for editing teacher values by passing a TeacherID
into the function

```

```

Public Function TeacherEditValues(ByVal TeacherID As String)
    'sets the SQL statement
    sql = "Select * FROM Teacher WHERE TeacherID = " & TeacherID & ";"
    'stores a SQL datadapter variable that connects to the connection and uses the
above SQL statement to query the SQL database
    Dim dataadapter As New SqlDataAdapter(sql, con)
    'stores a dataset variable
    Dim ds As New DataSet
    'The dataadapter fills the dataset with the result of the SQL query
    dataadapter.Fill(ds, "Teacher")
    'returns the dataset
    Return ds
End Function

'this function sets a variable GCSEweighting that determines what percentage a subject
point score is multiplied by
'the values of i (from the for next variable) and the datatable used for the grade
prediciton are passed into this function
Function GCSEGradewighting(ByVal i As Integer, ByVal dt As DataTable)
    'stores the weighting as a double so that it can represent a large floating point
number
    Dim GCSEweighting As Double

    'if a new subject has been added to the program then its weighting is set to 1
    GCSEweighting = 1

    'If the value of the cell on the row of i in the datatable under the heading of
    "subject" matches any of the following list of subjects then the GCSEweighting variable is
    set,
        'the weighting variable changes depending on the subject, to indicate the relative
    difficulty of the subject
        If dt.Rows(i)("Subject").ToString() = "Chemistry" Then
            GCSEweighting = 1.22
        End If
        If dt.Rows(i)("Subject").ToString() = "Physics" Then
            GCSEweighting = 1.19
        End If
        If dt.Rows(i)("Subject").ToString() = "Biology" Then
            GCSEweighting = 1.18
        End If
        If dt.Rows(i)("Subject").ToString() = "German" Or dt.Rows(i)("Subject").ToString()
= "French" Or dt.Rows(i)("Subject").ToString() = "Spanish" Then
            GCSEweighting = 1.12
        End If
        If dt.Rows(i)("Subject").ToString() = "Maths" Then
            GCSEweighting = 1.11
        End If
        If dt.Rows(i)("Subject").ToString() = "Computing" Then
            GCSEweighting = 1.08
        End If
        If dt.Rows(i)("Subject").ToString() = "History" Then
            GCSEweighting = 1.03
        End If
        If dt.Rows(i)("Subject").ToString() = "ICT" Then
            GCSEweighting = 1.03
        End If
        If dt.Rows(i)("Subject").ToString() = "Geography" Then
            GCSEweighting = 1
        End If
        If dt.Rows(i)("Subject").ToString() = "English Language" Or
dt.Rows(i)("Subject").ToString() = "English Literature" Then
            GCSEweighting = 0.98
        End If

```

```

    If dt.Rows(i)("Subject").ToString() = "Music" Then
        GCSEweighting = 0.96
    End If
    If dt.Rows(i)("Subject").ToString() = "Religious Studies" Then
        GCSEweighting = 0.94
    End If
    If dt.Rows(i)("Subject").ToString() = "Business Studies" Then
        GCSEweighting = 0.93
    End If
    If dt.Rows(i)("Subject").ToString() = "Art" Or dt.Rows(i)("Subject").ToString() =
"Graphics" Then
        GCSEweighting = 0.86
    End If
    If dt.Rows(i)("Subject").ToString() = "Textiles" Or
dt.Rows(i)("Subject").ToString() = "Health And Social care" Then
        GCSEweighting = 0.88
    End If
    If dt.Rows(i)("Subject").ToString() = "Resistant Materials" Or
dt.Rows(i)("Subject").ToString() = "Physical Education" Then
        GCSEweighting = 0.87
    End If
    If dt.Rows(i)("Subject").ToString() = "Leisure and Tourism" Or
dt.Rows(i)("Subject").ToString() = "Drama" Then
        GCSEweighting = 0.84
    End If

    'returns the variable GCSEweighting
    Return GCSEweighting
End Function

'this function returns a predicted grade as a letter
'the values of the subject and average pointscore are passed into this function, the
index of i in the for next statement is passed to this function and the datatable in use
is also passed
Function A2predictedgrade(ByVal subject As String, ByVal i As Integer, ByVal dt As
DataTable, ByVal averagepointscore As Integer)
    'Predictedgrade is stored as a string because it returns a non-numeric grade
    Dim predictedgrade As String
    'subject point score is stored as a double so that it can be a large floating
point number
    Dim subjectpointscore As Double

    'if a new subject has been added to the system then the subjectpointscore of that
subject is set as the averagepointscore
    subjectpointscore = averagepointscore

    'If the value of the cell on the row of i in the datatable under the heading of
"subject" matches any of the following list of subjects then the Subjectpointscore
variable is set,
    'the variable changes depending on the subject, to indicate the relative
difficulty of the subject, and equals the averagepointscore multiplied by the relative
difficulty weighting of that subject
    If dt.Rows(i)("Subject").ToString() = "Chemistry" Then
        subjectpointscore = averagepointscore * 0.83
    End If
    If dt.Rows(i)("Subject").ToString() = "Physics" Then
        subjectpointscore = averagepointscore * 0.85
    End If
    If dt.Rows(i)("Subject").ToString() = "Biology" Then
        subjectpointscore = averagepointscore * 0.85
    End If
    If dt.Rows(i)("Subject").ToString() = "German" Or dt.Rows(i)("Subject").ToString() =
"French" Or dt.Rows(i)("Subject").ToString() = "Spanish" Then

```

```

        subjectpointscore = averagepointscore * 0.87
    End If
    If dt.Rows(i)("Subject").ToString() = "Maths" Then
        subjectpointscore = averagepointscore * 0.89
    End If
    If dt.Rows(i)("Subject").ToString() = "Computing" Then
        subjectpointscore = averagepointscore * 0.9
    End If
    If dt.Rows(i)("Subject").ToString() = "History" Then
        subjectpointscore = averagepointscore * 0.91
    End If
    If dt.Rows(i)("Subject").ToString() = "ICT" Then
        subjectpointscore = averagepointscore * 0.95
    End If
    If dt.Rows(i)("Subject").ToString() = "Geography" Then
        subjectpointscore = averagepointscore * 0.93
    End If
    If dt.Rows(i)("Subject").ToString() = "English Language" Or
dt.Rows(i)("Subject").ToString() = "English Literature" Then
        subjectpointscore = averagepointscore * 0.94
    End If
    If dt.Rows(i)("Subject").ToString() = "Music" Then
        subjectpointscore = averagepointscore * 0.96
    End If
    If dt.Rows(i)("Subject").ToString() = "Religious Studies" Then
        subjectpointscore = averagepointscore * 0.99
    End If
    If dt.Rows(i)("Subject").ToString() = "Business Studies" Then
        subjectpointscore = averagepointscore * 1.01
    End If
    If dt.Rows(i)("Subject").ToString() = "Art" Or dt.Rows(i)("Subject").ToString() =
"Graphics" Then
        subjectpointscore = averagepointscore * 1.03
    End If
    If dt.Rows(i)("Subject").ToString() = "Textiles" Or
dt.Rows(i)("Subject").ToString() = "Health And Social care" Then
        subjectpointscore = averagepointscore * 1.02
    End If
    If dt.Rows(i)("Subject").ToString() = "Resistant Materials" Or
dt.Rows(i)("Subject").ToString() = "Physical Education" Then
        subjectpointscore = averagepointscore * 0.88
    End If
    If dt.Rows(i)("Subject").ToString() = "Leisure and Tourism" Or
dt.Rows(i)("Subject").ToString() = "Drama" Then
        subjectpointscore = averagepointscore * 0.97
    End If

'The value of the subjectpointscore determines the predicted grade variable
'for example if the subjectpoint score is 42, then it is lower than 46 but greater
than 40, which therefore returns a predicted grade of C
If subjectpointscore >= 58 Then
    predictedgrade = "A*"
ElseIf subjectpointscore < 58 And subjectpointscore >= 52 Then
    predictedgrade = "A"
ElseIf subjectpointscore < 52 And subjectpointscore >= 46 Then
    predictedgrade = "B"
ElseIf subjectpointscore < 46 And subjectpointscore >= 40 Then
    predictedgrade = "C"
ElseIf subjectpointscore < 40 And subjectpointscore >= 34 Then
    predictedgrade = "D"
ElseIf subjectpointscore < 34 And subjectpointscore >= 28 Then

```

```
predictedgrade = "E"
ElseIf subjectpointscore < 28 Then
    predictedgrade = "F"
End If

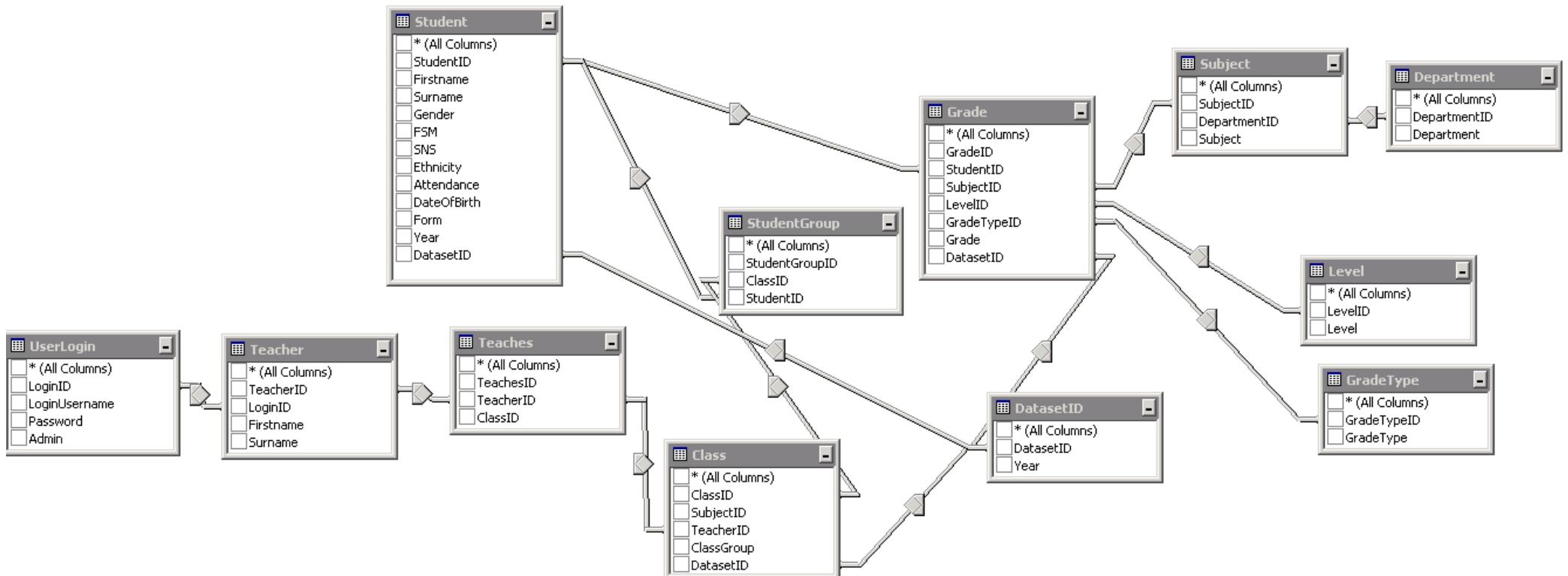
'returns the predictedgrade variable (the predicted grade that will be output to
the user for that given subject)
Return predictedgrade
End Function
End Module
```

Database Table (entity) Relationship Diagram

Since there are no many-many relationships present in the table relationship diagram below, it shows that the system is indeed in **third normal form**.

Key of table relationships:

One-One relationship	
One-many relationship or many-one relationship	
Many to many relationship	



Sample of Algorithms

The following sample of algorithms here are the ones that I designed accordingly in my test section and were algorithms I expected to have to implement. However, here I have included the actual name of the algorithms within my code, a location example of where the algorithm is used and the actual code I used to implement the algorithm. The pseudo-codes and descriptions of the algorithms remain the same as the ones in the design section.

Algorithm:	Login_Click	Location:	Login
Description			
This algorithm will be used to determine whether a valid user has or has not attempted to log into the system. It will first check whether the entered details are valid and then it will check whether the logged in user is an admin or not. The user will then be given a set of privileges based on their admin status. If the username or password entered is not valid then the fields should be reset to blank.			
Pseudo-code			
<pre> Dim sqlquery Dim username Dim password Dim datatable Dim admin Sqlquery.connection=con Sqlquery = "SELECT LoginUsername, Password FROM UserLogin WHERE LoginUsername = "" & usernamefield.Text & "" AND Password = "" & passwordfield.Text & "" Datatable.fill USING sqlquery If rows = true THEN Password=check.datatable.(“password”) username=check.datatable.(“username”) admin=check.datatable.(“admin”) DISPLAY mainmenu ELSE Output (“Invalid Username or Password”) Usernamefield.text = “” Passwordfield.text = “” End if </pre>			
Code			
<pre> Private Sub Login_Click(sender As System.Object, e As System.EventArgs) Handles Loginbutton.Click Dim Cmd As New SqlCommand Dim da As New SqlDataAdapter Dim dt As New DataTable Dim ds As New DataSet Dim password As String Dim Username As String Dim con As New SqlConnection Dim sqlquery As New SqlCommand </pre>			

```
Dim username1 As String
con.ConnectionString = "Data Source=<>;Initial Catalog=<>;User
ID=<>;Password=<>"

Try
    con.Open()
Catch ex As Exception
    MsgBox("Can not open SQL Server connection! ")
End Try
sqlquery.Connection = con
sqlquery.CommandText = "SELECT LoginUsername, Password FROM UserLogin WHERE
LoginUsername = '" & txtUsername.Text & "' AND Password = '" & txtPassword.Text & "'"
Dim check As SqlDataReader = sqlquery.ExecuteReader()

If check.HasRows Then
    While check.Read()
        password = check("Password").ToString()
        Username = check("LoginUsername").ToString()
        If Username = check("LoginUsername") Then
            username1 = Username
        End If
        modulepassword = txtPassword.Text
        moduleusername = txtUsername.Text
        Main_Menu.Show()
        Me.Close()
    End While
    con.Close()
    check.Dispose()
    DatabaseConnect()
    sql = "SELECT DISTINCT * FROM UserLogin WHERE LoginUsername = '" &
moduleusername & "' AND Password = '" & modulepassword & "'"
    Cmd.Connection = con
    Cmd.CommandText = sql
    da.SelectCommand = Cmd
    da.Fill(ds, "temp")
    moduleuserid = ds.Tables(0).Rows(0).Item("LoginID").ToString()
Else
    MessageBox.Show("Invalid Username or Password", "Authentication Failure",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
    txtPassword.Text = ""
    txtUsername.Text = ""
End If
End Sub
```

Algorithm: Field_SelectedIndexChanged	Location: Add Student Class
Description	
This algorithm is used to display different objects on the form depending on the Case of a field as it is changed. When the Case value Matches the field value then the corresponding Case runs its code. For example if the field.text value is "Name" then all code within the name case will run.	
Pseudo-code	
<pre> Procedure handles field.indexchanged Select Case "Selected-Field" Case "" Namefield = non-visible Formfield = non-visible Yearfield= non-visible displayStudentgroup = non-visible Valuelabel = non-visible Case "Name" Namefield = Visible Formfield = non-visible Yearfield= non-visible displayStudentgroup = Visible Valuelabel = Visible Valuelabel.text = "Name :" Case "Form" Generatecombobox(Searchform) Using ("Student", "Form") Namefield = non-visible Formfield = Visible Yearfield= non-visible displayStudentgroup = Visible Valuelabel = Visible Valuelabel.text = "Form :" End Select End Procedure </pre>	
code	
<pre> Select Case Field.Text Case "" SearchName.Visible = False SearchForm.Visible = False SearchYear.Visible = False Displaystudent.Visible = False wherelabel.Visible = False Case "Name" SearchName.Visible = True SearchForm.Visible = False SearchYear.Visible = False Displaystudent.Visible = True wherelabel.Text = "Name:" Case "Form" dt = GenerateCombo("Student", "Form", False) SearchForm.DataSource = dt SearchForm.DisplayMember = "Form" SearchName.Visible = False </pre>	

```
SearchForm.Visible = True
SearchYear.Visible = False
Displaystudent.Visible = True
wherelabel.Text = "Form:"
Case "Year"
    dt = GenerateCombo("Student", "Year", False)
    SearchYear.DataSource = dt
    SearchYear.DisplayMember = "Year"
    SearchName.Visible = False
    SearchForm.Visible = False
    SearchYear.Visible = True
    Displaystudent.Visible = True
    wherelabel.Text = "Year:"
End Select
```

Algorithm: DatabaseConnect	Location: Module1
Description	
This algorithm connects the system to the SQL Server database through a connection string that is set by the program. This connection string allows a SQL connection to be made that uses the login credentials of the connection string to access the database. The algorithm first attempts to establish a connection using the connection string which has been set, if a connection cannot be established then an appropriate message is displayed to the user.	
Pseudo-code	
<pre> Procedure DatabaseConnect() Imports System.SQL Procedure Databaseconnect () Dim Connectionstring Dim con Dim sql Dim connection Connection string = "Database Connection Path here" Con = SQLconnection USING connectionstring Try Con.open Catch exception Output("Can not open SQL Server connection! Make Sure Server Device is Switched on. ") End try End Procedure </pre>	
code	
<pre> Public connectionString As String Public con As SqlConnection Public sql As String Dim connection As New SqlConnection connectionstring = "Data Source=<>;Initial Catalog=<>;User ID=<>;Password=<>" con = New SqlConnection(connectionString) Try con.Open() Catch ex As Exception MsgBox("Can not open SQL Server connection! Make Sure Server Device is Switched on. ") End Try </pre>	

Algorithm: GenerateCombo	Location: Module1
Description	
This algorithm is set as a function so that a dynamic table, field and needemptyvariables can be passed into the function. Once a table and field are passed into the function a SQL statement is used to return records using the table and field values. These records are then stored within a dataset table. The dataset table is then set as the filterdt property, which is the data table returned once the function has been processed. If the Boolean state of needempty is true then a new row is inserted at the row position 0 to the filterdt data table.	
Pseudo-code	
<pre> Procedure GenerateCombo(table, field, needempty) Sql = "SELECT DISTINCT " & field & " FROM " & table & " ORDER BY " & field & " ASC" Dim FilterDS Dim FilterDT Datadapter = New SQLdataadaptor USING (sql,con) Datadapter.fill(FilterDS, table) Filterdt= Filterds If needempty = true then Insert New Row at filterDt.rows(0) End If Return FilterDT End Procedure </pre>	
code	
<pre> Function GenerateCombo(table As String, field As String, needEmpty As Boolean) sql = "SELECT DISTINCT " & field & " FROM " & table & " ORDER BY " & field & " ASC" Dim dataadapter As New SqlDataAdapter(sql, con) Dim FilterDS As New DataSet Dim FilterDT As DataTable dataadapter.Fill(FilterDS, table) FilterDT = FilterDS.Tables(0) If needEmpty Then Dim drNewRow As DataRow = FilterDT.NewRow FilterDT.Rows.InsertAt(drNewRow, 0) FilterDT.AcceptChanges() End If Return FilterDT End Function </pre>	

Algorithm: GCSEGradeWeighting	Location: Module1
Description	
This algorithm Sets the grade weighting of a GCSE subject and returns the value of the grade weighting so that it can be applied to the corresponding grade. These weightings are particularly important because they are used to calculate a varied range of realistic grade predictions based on the relative difficulties of the subjects the student has taken at GCSE level.	
Pseudo-code	
<pre> Procedure GCSEWeighting(l,dt) Dim GCSEweighting If dt.rows(i)(“subject”) = “Chemistry” Then GCSEweighting = 1.22 End if If dt.rows(i)(“subject”) = “Physics” Then GCSEweighting = 1.19 End if If dt.rows(i)(“subject”) = “Biology” Then GCSEweighting = 1.18 End if ... Return GCSEweighting End Procedure </pre>	
code	
<pre> Function GCSEGradewighting(ByVal i As Integer, ByVal dt As DataTable) Dim GCSEweighting As Double GCSEweighting = 1 If dt.Rows(i)(“Subject”).ToString() = “Chemistry” Then GCSEweighting = 1.22 End If If dt.Rows(i)(“Subject”).ToString() = “Physics” Then GCSEweighting = 1.19 End If If dt.Rows(i)(“Subject”).ToString() = “Biology” Then GCSEweighting = 1.18 End If If dt.Rows(i)(“Subject”).ToString() = “German” Or dt.Rows(i)(“Subject”).ToString() = “French” Or dt.Rows(i)(“Subject”).ToString() = “Spanish” Then GCSEweighting = 1.12 End If If dt.Rows(i)(“Subject”).ToString() = “Maths” Then GCSEweighting = 1.11 End If If dt.Rows(i)(“Subject”).ToString() = “Computing” Then GCSEweighting = 1.08 End If If dt.Rows(i)(“Subject”).ToString() = “History” Then GCSEweighting = 1.03 End If If dt.Rows(i)(“Subject”).ToString() = “ICT” Then GCSEweighting = 1.03 End If If dt.Rows(i)(“Subject”).ToString() = “Geography” Then GCSEweighting = 1 End If If dt.Rows(i)(“Subject”).ToString() = “English Language” Or dt.Rows(i)(“Subject”).ToString() = “English Literature” Then </pre>	

```
    GCSEweighting = 0.98
End If
If dt.Rows(i)("Subject").ToString() = "Music" Then
    GCSEweighting = 0.96
End If
If dt.Rows(i)("Subject").ToString() = "Religious Studies" Then
    GCSEweighting = 0.94
End If
If dt.Rows(i)("Subject").ToString() = "Business Studies" Then
    GCSEweighting = 0.93
End If
If dt.Rows(i)("Subject").ToString() = "Art" Or
dt.Rows(i)("Subject").ToString() = "Graphics" Then
    GCSEweighting = 0.86
End If
If dt.Rows(i)("Subject").ToString() = "Textiles" Or
dt.Rows(i)("Subject").ToString() = "Health And Social care" Then
    GCSEweighting = 0.88
End If
If dt.Rows(i)("Subject").ToString() = "Resistant Materials" Or
dt.Rows(i)("Subject").ToString() = "Physical Education" Then
    GCSEweighting = 0.87
End If
If dt.Rows(i)("Subject").ToString() = "Leisure and Tourism" Or
dt.Rows(i)("Subject").ToString() = "Drama" Then
    GCSEweighting = 0.84
End If
Return GCSEweighting
End Function
```

Algorithm: A2PredictedGrade	Location: Module1
Description	
This function is used to generate a unique A2 predicted grade for a subject depending on a student's average point score and the subject the grade is predicted for. A weighting value is applied to the average point score of a student to determine the predicted grade of an A2 subject depending on its relative difficulty. Once the point score has been calculated, it is then processed by a second algorithm within this function which determines the actual A2 predicted grade by converting the subject score to a grade. This varies depending on the point score so that a range of grades are achieved depending on the average point score of the student.	
Pseudo-code	
<pre> Procedure A2Predictedgrade(subject, i, dt, averagepointscore) Dim predictedgrade Dim subjectpointscore Subjectpointscore=averagepointscore If dt.rows(i)(“subject”) = “Chemistry” Then Subjectpointscore = averagepointscore * 0.83 End if If dt.rows(i)(“subject”) = “Physics” Then Subjectpointscore = averagepointscore * 0.85 End if If dt.rows(i)(“subject”) = “Biology” Then Subjectpointscore = averagepointscore * 0.85 End if ... If subjectpointscore >= 58 then Predictedgrade = “A*” Elseif subjectpointscore < 58 and subjectpointscore >= 52 then Predictedgrade = “A” Elseif subjectpointscore < 52 and subjectpointscore >= 46 then Predictedgrade = “B” Elseif subjectpointscore < 46 and subjectpointscore >= 40 then Predictedgrade = “C” Elseif subjectpointscore < 40 and subjectpointscore >= 34 then Predictedgrade = “D” Elseif subjectpointscore < 34 and subjectpointscore >= 28 then Predictedgrade = “E” Elseif subjectpointscore < 28 then Predictedgrade = “U” End if Return Predictedgrade End Procedure </pre>	
code	
<pre> Function A2predictedgrade(ByVal subject As String, ByVal i As Integer, ByVal dt As DataTable, ByVal averagepointscore As Integer) Dim predictedgrade As String Dim subjectpointscore As Double subjectpointscore = averagepointscore If dt.Rows(i)(“Subject”).ToString() = “Chemistry” Then subjectpointscore = averagepointscore * 0.83 End If If dt.Rows(i)(“Subject”).ToString() = “Physics” Then subjectpointscore = averagepointscore * 0.85 </pre>	

```
End If
If dt.Rows(i)("Subject").ToString() = "Biology" Then
    subjectpointscore = averagepointscore * 0.85
End If
If dt.Rows(i)("Subject").ToString() = "German" Or
dt.Rows(i)("Subject").ToString() = "French" Or dt.Rows(i)("Subject").ToString() =
"Spanish" Then
    subjectpointscore = averagepointscore * 0.87
End If
If dt.Rows(i)("Subject").ToString() = "Maths" Then
    subjectpointscore = averagepointscore * 0.89
End If
If dt.Rows(i)("Subject").ToString() = "Computing" Then
    subjectpointscore = averagepointscore * 0.9
End If
If dt.Rows(i)("Subject").ToString() = "History" Then
    subjectpointscore = averagepointscore * 0.91
End If
If dt.Rows(i)("Subject").ToString() = "ICT" Then
    subjectpointscore = averagepointscore * 0.95
End If
If dt.Rows(i)("Subject").ToString() = "Geography" Then
    subjectpointscore = averagepointscore * 0.93
End If
If dt.Rows(i)("Subject").ToString() = "English Language" Or
dt.Rows(i)("Subject").ToString() = "English Literature" Then
    subjectpointscore = averagepointscore * 0.94
End If
If dt.Rows(i)("Subject").ToString() = "Music" Then
    subjectpointscore = averagepointscore * 0.96
End If
If dt.Rows(i)("Subject").ToString() = "Religious Studies" Then
    subjectpointscore = averagepointscore * 0.99
End If
If dt.Rows(i)("Subject").ToString() = "Business Studies" Then
    subjectpointscore = averagepointscore * 1.01
End If
If dt.Rows(i)("Subject").ToString() = "Art" Or
dt.Rows(i)("Subject").ToString() = "Graphics" Then
    subjectpointscore = averagepointscore * 1.03
End If
If dt.Rows(i)("Subject").ToString() = "Textiles" Or
dt.Rows(i)("Subject").ToString() = "Health And Social care" Then
    subjectpointscore = averagepointscore * 1.02
End If
If dt.Rows(i)("Subject").ToString() = "Resistant Materials" Or
dt.Rows(i)("Subject").ToString() = "Physical Education" Then
    subjectpointscore = averagepointscore * 0.88
End If
If dt.Rows(i)("Subject").ToString() = "Leisure and Tourism" Or
dt.Rows(i)("Subject").ToString() = "Drama" Then
    subjectpointscore = averagepointscore * 0.97
End If
If subjectpointscore >= 58 Then
    predictedgrade = "A*"
ElseIf subjectpointscore < 58 And subjectpointscore >= 52 Then
    predictedgrade = "A"
ElseIf subjectpointscore < 52 And subjectpointscore >= 46 Then
    predictedgrade = "B"
ElseIf subjectpointscore < 46 And subjectpointscore >= 40 Then
    predictedgrade = "C"
ElseIf subjectpointscore < 40 And subjectpointscore >= 34 Then
```

```
    predictedgrade = "D"
    ElseIf subjectpointscore < 34 And subjectpointscore >= 28 Then
        predictedgrade = "E"
    ElseIf subjectpointscore < 28 Then
        predictedgrade = "F"
    End If
    Return predictedgrade
End Function
```

Algorithm: Txtvalidate	Location: Edit Classes
Description	
This function validates whether a set of textbox or combo box entry fields are not empty and that they have values selected by the user. If all fields have been completed then the function returns a true state. However if any field has not been completed then a false state is returned. Also fields that have not been completed have their background colours changed to a red colour to indicate this to the user. After a field has been completed its background is returned to a white colour.	
Pseudo-code	
<pre> Procedure txtvalidate() Txtvalidate =true If firstnamefield. Text = "" Then Firstnamefield.background = red Txtvalidate =false Else Firstnamefield.background = white End if If surnamefield. Text = "" Then surnamefield.background = red Txtvalidate =false Else surnamefield.background = white End if If classfield. Text = "" Then classfield.background = red Txtvalidate =false Else classfield.background = white End if ... End Procedure </pre>	
code	
<pre> Private Function txtvalidate() As Boolean Call DatabaseConnect() txtvalidate = True If Firstnamefield.Text = "" Then Firstnamefield.BackColor = Color.Salmon txtvalidate = False Else Firstnamefield.BackColor = Color.White End If If Surnamefield.Text = "" Then Surnamefield.BackColor = Color.Salmon txtvalidate = False Else Surnamefield.BackColor = Color.White End If If Classgroupfield.Text = "" Then Classgroupfield.BackColor = Color.Salmon </pre>	

```
    txtvalidate = False
Else
    Classgroupfield.BackColor = Color.White
End If
If Subjectfield.Text = "" Then
    Subjectfield.BackColor = Color.Salmon
    txtvalidate = False
Else
    Subjectfield.BackColor = Color.White
End If
If Yearfield.Text = "" Then
    Yearfield.BackColor = Color.Salmon
    txtvalidate = False
Else
    Yearfield.BackColor = Color.White
End If
End Function
```

Algorithm: btnExport_Click	Location: Grade_Predictions
Description	
This algorithm is responsible for building a CSV formatted string using the contents of a data grid that is being displayed on a form. Once the CSV style string has been written, a user is then able to select where to save a file and is also given the option of naming the file. From here the CSV formatted string is written to the new file the user has named in the desired location the user has set.	
Pseudo-code	
Procedure ExportCSV() Dim csv For each datagridviewcolumn in "datagrid".columns Csv += column(headertext) & "," Next Csv.new line For each datagridviewrow in "datagrid".rows For each datagridviewcell in "datagrid".rows.cells Csv += cell.text & "," ";" & "," Next Csv.new line Next Dim savefiledialog As save filedialog() svaefiledialog.filterselection = "Spreadsheet CSV file (*.csv) *.csv" savefieldialog.restoredirectory = true if savefiledialog.result = ok then Dim SW as streamwriter using savefiledialog.openfile If SW is not Nothing then Sw.write(csv) Sw.close() End if End if	
code	
<pre> Dim csv As String = String.Empty For Each column As DataGridViewColumn In grouppredictionsgrid.Columns csv += column.HeaderText & ","c Next csv += vbCr & vbLf For Each row As DataGridViewRow In grouppredictionsgrid.Rows For Each cell As DataGridViewCell In row.Cells csv += cell.Value.ToString().Replace(",", ";") & ","c Next csv += vbCr & vbLf Next Dim saveFileDialog1 As New SaveFileDialog() saveFileDialog1.Filter = "Spreadsheet CSV file (*.csv) *.csv" saveFileDialog1.FilterIndex = 1 saveFileDialog1.RestoreDirectory = True </pre>	

```
If saveFileDialog1.ShowDialog() = DialogResult.OK Then
    Dim sw As StreamWriter = New StreamWriter(saveFileDialog1.OpenFile())
    If (sw IsNot Nothing) Then
        sw.WriteLine(csv)
        sw.Close()
    End If
End If
```

Algorithm: importcsv_Click	Location: Student
Description	
With this algorithm a user will be able to select an academic year and then import a CSV formatted spread sheet file into the database system. When it came to implementation I added validation functionality which would indicate for each row whether it had or had not been successfully added to the SQL server database. I outlined in my design of this algorithm that I may consider doing this.	
Pseudo-code	
<pre> Dim openfiledialog Dim dataset Dim filepath Dim datagrid Dim datatable Dim sqlquery Dim rows Dim x If openfiledialog.selection = ok Then Filepath = GET FILE PATH Dim connectionstring USING filepath AND USING Delimiter OR Splitstring Connectionstring.open Sqlquery = "SELECT * FROM " & filepath & " Datagrid.fill USING sqlquery Datagrid.addnewcolumn = ("inserted (true/false)") Rows = datagrid.rowcount X=0 For datagrid.rowcount = 0 to datagrid.rowcount = rows Sqlquery = " INSERT INTO Student Values (datagrid.row(x)(0), datagrid.row(x)(1) ... datagrid.row(x)(9), X = x + 1 Next Output ("Insert Data successful") Else Ouptut ("please select an academic year to import student details for") End if </pre>	
Code	
<pre> If academicyearcombo.SelectedValue <> "0" Then Dim dlg As New OpenFileDialog Dim ds As New DataSet Dim path As String = "My Filename" Dim path1 As String Dim dt As New DataTable Dim dt1 As New DataTable Dim griddt As New DataTable Dim rowadded As String Dim da As New SqlDataAdapter </pre>	

```

Dim Sid As Integer

dlg.Filter = "CSV Files (*.csv)|Any Files (*.*)"
If dlg.ShowDialog = System.Windows.Forms.DialogResult.OK Then
    path = dlg.FileName
    path1 = dlg.FileName
    path = System.IO.Path.GetDirectoryName(path)
    path1 = System.IO.Path.GetFileName(path1)
    Dim strConnString As String = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=" & path & ";Extended Properties=Text;"
    Dim conn As New OleDbConnection(strConnString)
    conn.Open()
    Dim cmd As New OleDbCommand("SELECT * FROM [" & path1 & "]", conn)
    Dim da1 As New OleDbDataAdapter()
    da1.SelectCommand = cmd
    da1.Fill(griddt)
    da1.Dispose()
    conn.Close()
    griddt.Columns.Add("Inserted (True/False)")
    Try
        Dim x As Integer = 0
        Dim csvrows As Integer
        csvrows = griddt.Rows.Count
        sql = "SELECT TOP 1 StudentID FROM Student ORDER BY StudentID
DESC"
        Dim cmmnd As New SqlCommand(sql, con)
        da.SelectCommand = cmmnd
        da.Fill(ds, "Student")
        Sid = ds.Tables(0).Rows(0).Item("StudentID").ToString() + 1

        While (x <= csvrows - 1)
            Dim originaldatabaserows As Integer
            Dim afterinsertdatabaserows As Integer
            sql = "SELECT * From Student"
            Dim sqlcommand As New SqlCommand(sql, con)
            da.SelectCommand = sqlcommand
            da.Fill(ds, "temp")
            dt = ds.Tables("temp")
            originaldatabaserows = dt.Rows.Count

            sql = "INSERT INTO Student VALUES(@SID, @Fname, @Sname,
@Gender, @FSM, @SNS, @Ethnicity, @Attendance, @DOB, @Form, @Year, @DatasetID) SET
ANSI_WARNINGS OFF"
            Using cmd1 = New SqlCommand(sql, con)
                cmd1.Parameters.AddWithValue("SID", Sid)
                cmd1.Parameters.AddWithValue("Fname", griddt.Rows(x)(0))
                cmd1.Parameters.AddWithValue("Sname", griddt.Rows(x)(1))
                cmd1.Parameters.AddWithValue("Gender", griddt.Rows(x)(2))
                cmd1.Parameters.AddWithValue("FSM", griddt.Rows(x)(3))
                cmd1.Parameters.AddWithValue("SNS", griddt.Rows(x)(4))
                cmd1.Parameters.AddWithValue("Ethnicity",
griddt.Rows(x)(5))
                cmd1.Parameters.AddWithValue("Attendance", dt.Rows(x)(6))
                cmd1.Parameters.AddWithValue("DOB", griddt.Rows(x)(7))
                cmd1.Parameters.AddWithValue("Form", griddt.Rows(x)(8))
                cmd1.Parameters.AddWithValue("Year", griddt.Rows(x)(9))
                cmd1.Parameters.AddWithValue("DatasetID",
academicyearcombo.SelectedValue)
                cmd1.ExecuteNonQuery()
            End Using
            sql = "SELECT * From Student"
            Dim sqlcommand2 As New SqlCommand(sql, con)

```

```
        da.SelectCommand = sqlcommand
        da.Fill(ds, "temp1")
        dt1 = ds.Tables("temp1")
        afterinsertdatabaserows = dt1.Rows.Count
        If (afterinsertdatabaserows) - (originaldatabaserows) = (1 +
x) Then
            rowadded = "True"
        Else
            rowadded = "False"
        End If
        griddt.Rows(x)("Inserted (True/False)") = rowadded
        x = x + 1
        Sid = Sid + 1
    End While
    MsgBox("Insert Data Successful", MsgBoxStyle.Information, "Data
Upload")
    dt = Nothing
    StudentGrid.DataSource = griddt
    StudentGrid.Visible = True
    StudentGrid.DataMember = ""
    StudentGrid.AutoResizeColumns()
    Catch ex As Exception
        MsgBox(ex.Message)
    End Try
End If
Else
    MsgBox("Please Select an Academic Year ", MsgBoxStyle.Information,
MessageBoxButtons.OK)
End If
End Sub
```

Sample of SQL Queries

1) SQL Query to select a set of fields from one table

```
Select StudentID, Firstname, Surname, Gender, FSM, SNS, Ethnicity,
Attendance, DateOfBirth, Form, Year
FROM Student
```

2) SQL query to select non-repeating records from one table and order them in ascending alphabetic order

```
SELECT DISTINCT *
FROM Level
ORDER BY Level ASC
```

3) SQL Query to select a distinct set of records from more than one table and using a variable as the value of the WHERE command

```
SELECT DISTINCT Class.ClassGroup, Subject.Subject, DatasetID.Year
FROM Department INNER JOIN Subject ON Department.DepartmentID =
Subject.DepartmentID INNER JOIN Class INNER JOIN DatasetID ON
Class.DatasetID = DatasetID.DatasetID ON Subject.SubjectID =
Class.SubjectID INNER JOIN Teacher INNER JOIN Teaches ON Teacher.TeacherID =
Teaches.TeacherID ON Class.TeacherID = Teacher.TeacherID
WHERE Class.ClassID = " & ClassID & "
```

4) SQL query to insert a new record into the database using a series of parameters as the values which are inserted

```
INSERT INTO STUDENT(StudentID, Firstname, Surname, Gender, FSM, SNS,
Ethnicity, Attendance, DateOfBirth, Form, Year, DatasetID) VALUES
(@StudentID, @Fname, @Sname, @Gender, @Meals, @SNS, @Ethnicity,
@Attendance, @Birth, @Form, @Year, @Dataset)
```

5) SQL Query to update an existing database record using set parameters and where a field matches a variable text value

```
UPDATE UserLogin SET LoginUsername = @LoginUsername, Password = @Password,
Admin = @Admin WHERE LoginID = '" & loginidfield.Text & "'
```

6) SQL query to select the top record from a query after ordering records in descending order of ID value

```
Select TOP 1 StudentGroupID
FROM StudentGroup
ORDER BY StudentGroupID DESC
```

7) SQL Query which concatenates one or more SQL strings together depending on the value of a field, using a case statement.

```
sql = " Select Firstname, Surname, Form, Year FROM Student "
```

```
Select Case Field.Text
    Case "Name"
        sql = sql & " WHERE Firstname LIKE '%" & SearchName.Text &
        "%' Or Surname LIKE '%" & SearchName.Text & "%' ORDER BY StudentID ASC "
    Case "Form"
        sql = sql & " WHERE Form LIKE '%" & SearchForm.Text &
        "%' ORDER BY StudentID ASC "
    Case "Year"
        sql = sql & " WHERE Year LIKE '%" & SearchYear.Text &
        "%' ORDER BY StudentID ASC "
End Select
```

- 8) SQL Query that deletes a record from the database that matches one or more variables**

```
DELETE FROM Teacher WHERE Teacher.TeacherID = '' & TID & '' AND Firstname = '' & Firstname.Text & '' and Teacher.Surname = '' & Surname.Text & '';
```

- 9) SQL Query that uses a pivot table to create a custom set of columns that can be used to count the number of grades underneath a column by either placing a 1 Or a 0 to indicate that the grade of that row is present. A 1 represents that the grade is under that column e.g. a grade under the b column would be a 1 and all other columns would have a 0 underneath their column to indicate that there is not a grade present.**

```
SELECT [A*],[A],[B],[C],[D],[E],[F],[G] FROM Grade PIVOT (Count(GradeID) FOR Grade IN ([A*],[A],[B],[C],[D],[E],[F],[G])) AS PVTable WHERE SubjectID = '' & SubjectField.SelectedValue & '' and DatasetID = '' & Yearfield.SelectedValue & '' and LevelID = '' & Levelevelfield.SelectedValue & '' and GradeTypeID = '' & Gradetypefield.SelectedValue & ''
```

- 10) SQL query that uses left joins as well as inner joins to select which fields are linked to which tables, a left join means that a one to many relationship is formed so that just one table inherits the fields of another table**

```
SELECT DISTINCT Grade.GradeID, Student.Firstname, Student.Surname, Subject.Subject, Grade.Grade, [Level].[Level], GradeType.GradeType, DatasetID.Year FROM Department INNER JOIN Subject ON Department.DepartmentID = Subject.DepartmentID CROSS JOIN Student LEFT OUTER JOIN StudentGroup INNER JOIN Class ON StudentGroup.ClassID = Class.ClassID INNER JOIN Grade ON Class.SubjectID = Grade.SubjectID INNER JOIN DatasetID ON Class.DatasetID = DatasetID.DatasetID INNER JOIN [Level] ON Grade.LevelID = [Level].LevelID INNER JOIN GradeType ON Grade.GradeTypeID = GradeType.GradeTypeID ON Student.StudentID = Grade.StudentID AND Student.StudentID = StudentGroup.StudentID WHERE Class.ClassID = '' & ClassID & '' AND level.Level = 'KS4' AND Subject.Subject = '' & subject & '' ORDER BY Student.Firstname ASC, Student.Surname ASC"
```

Testing

The testing of my system will refer to the test strategy and planning that I have done in my design section, however since developing the system I have now had to expand my testing quite dramatically as there are lots of specific components that require testing. Within my testing I will show whether the test is using typical, erroneous or exceptional data (boundary data) (TEX) and I will use the letters T, E and X to indicate which type of data I am testing. If any problems are located during my testing I will fix the bug if possible and then re-test the problem. If fixing the problem is not possible due to time constraints I will discuss the nature of any problems in my Testing Evaluation. Where possible I will provide detailed evidence of all tests carried out, these will be referenced per case within the **Actual Outcome** column and then shown by screenshots in my testing evidence section.

Login Form

Test No.	Input Data/Action	Expected Outcome	Actual Outcome	Comments and Actions
1	T: Test what happens when entering either a valid admin account's username and password or a standard teacher's username and password.	I expect the User to be directed to the Main Menu form and the system to validate if a user is admin or non-admin and appropriately show the correct main menu view.	As expected. Reference: 1	None required.
	E: Test what happens when entering a username or password that is incorrect. Whilst also testing what happens when a valid username is used with a non-valid password.	I expect the user to remain on the login form and the values of the text fields to be reset to blank. Also an error message should be displayed.	As expected.	None Required.
	X: Test what happens when either of the text fields are left blank and the user attempts to login.	The user should remain on the Login form and the incorrect user message should display.	As expected. Reference:2	None Required.
2	Test what happens when a user log's out from the main menu form.	The values of the Username and password fields should be reset to blank when a user returns to the login form.	As expected. Reference: 3	None Required.

Student Form

Test No.	Input Data/Action	Expected Outcome	Actual Outcome	Comments and Actions
3	T: Filter student selection by gender and display all male students.	I expect the data grid on the form to only be filled with male students.	As expected.	None Required.
4	T: Filter student selection on the form by typing the first name of a student into the value selection box. Using the first name "Will"	I expect the data grid to only return student records where either the first name or surname is similar to "Will"	As expected. Reference: 4	None Required.
	E: Test what happens when an invalid name is entered by the user into the filter selection. Test using the Invalid name "ifhejyp"	I expect the data grid to be filled with zero rows and an error message to display to the user.	The data grid appears when the error message is displayed but when the user selects ok, the data grid disappears. Reference: 5	None Required.
	X: Test what happens when a user does not enter a name into the filter selection but nevertheless clicks the search button.	I expect all student records for the chosen dataset year to be selected in the data grid.	As expected.	None Required.
5	T: Filter a student selection on the form by filtering by form group. I will attempt to filter all students within my system using the form "13L".	I expect the data grid to only be populated with students within the form group "13L".	As expected.	None Required.
6	Test what happens when a button within the data grid is clicked. In this instance it will be the "Edit Details" button.	I expect a second form to pop-up/display (the EditAdd Student form) and the values of the row the user wants to edit to be passed to this form.	As expected. Reference: 6	None Required.
7	Test what happens when the Add Student button is clicked by the user.	I expect a second form to pop-up/display (the EditAdd Student form) and the values of all fields on the form to be empty, ready for a new record to be inserted. I also expect	As expected. Reference: 7	None Required.

		the titles to display 'Add' text properties rather than 'edit' test properties.		
8	See what happens when the Add a new dataset year button is clicked by the user.	I expect the user to be redirected to the developer features form and for the developer features form to display with a selection of all current dataset years.	As expected. Reference: 8	None Required.
9	T: Test what happens when the user clicks the "Import CSV" button and selects an academic year to import data for with a valid CSV file format.	An open file dialog option should display which allows the user to select a CSV file to be imported. The results of which should then be displayed in a data grid with a column indicating whether the record has been successfully inserted into the database or not.	As expected. Reference: 9	None Required.
	E: Test what happens when a user attempts to import a CSV file but the file is already open within another program such as Microsoft Excel.	I expect an error message to be displayed saying that the file is already open.	As expected. Reference: 10	None Required.
	X: Test what happens when a user attempt to import a CSV file without choosing an academic year.	I expect an appropriate error message to display indicating this error.	As expected.	None Required.

Edit/Add Student Form

Test No.	Input Data/Action	Expected Outcome	Actual Outcome	Comments and Actions
10	T: Test to see what happens when attempting to insert a new student record using a valid set of data	The record should be added to the database with the details the user has entered. A new student ID should be assigned to the new record. A message should display indicating that the record has been added.	As expected. Reference 11:	None Required.
	X: test what happens when a user does not complete one or more textboxes.	An error message should display saying that "one or more fields have not been completed" and the background colour of any empty fields should become	As expected. Reference 12	None Required.

		red.		
	X: test what happens when a user completes previously incomplete textboxes but then clears another textbox that did have information previously entered and attempts to insert	An error message should display saying that “one or more fields have not been completed”. The textboxes that have now been completed should have their backgrounds changed back to white and the textbox that has now been cleared should have its background become red.	As expected. Reference 12	None Required.
11	T: test what happens when a user attempts to delete a valid record.	A message should initially display asking if the user really wants to delete the record. I expect the record the user has selected to be deleted from the database and a confirmation message to display.	As expected. Reference: 13	None Required.
	E: test what happens when a user attempts to delete a record that does not exist. I will clear the surname field on the form to test this.	A message should initially display asking if the user really wants to delete the record. I expect an error message to display indicating that the student record the user has entered does not exist in the database.	As expected. Reference: 14	None Required.
12	T: Test what happens when a user attempts to navigate away from the page but details on the form have not been edited.	I expect the form to close immediately and the user to be re-directed to the form they want to access.	As expected.	None Required.
	E: test what happens when navigating away from the form but values on the form have been edited.	I expect a confirmation message to display asking the user if they really want to navigate away from the page.	As expected. Reference: 15	None Required.

Teacher Form

Test No.	Input Data/Action	Expected Outcome	Actual Outcome	Comments and Actions
13	T: Test what happens when a user attempts to filter their teacher selection using a valid name. ("rees")	I expect the data grid to return the teacher records that match the name the user has entered.	As expected.	None Required.
	E: Test what happens when a non-valid name is entered into the teacher filter selection.	I expect the data grid to not display any teacher records and for a message box to display asking the user to input a valid teacher name.	As expected.	None Required.
	X: Test what happens when a user does not type any information into the teacher filter selection box but still clicks search.	I expect all teacher rows to be displayed in the data grid.	As expected. Reference: 16	None Required.
14	Test what happens when the user clicks the Add teacher button.	The EditAdd teacher form should display with all fields empty and all titles on the field displaying "add" text properties.	As expected.	None Required.
15	Test what happens when a button on the data grid is clicked. In this case the "Edit details" button.	The EditAdd teacher form should display with the values of the first name and surname textboxes completed with the same values as the row the user clicked to edit.	As expected. Reference: 17	None Required.
16	Test what happens when the print icon is clicked by the user	I expect a print dialog box to appear which will allow the user to select where to print a copy of the form and how many copies etc. the form should then be printed by the designated printer in landscape orientation.	As expected. Reference: 18	None Required.

Edit/Add Teacher Form

Test No.	Input Data/Action	Expected Outcome	Actual Outcome	Comments and Actions
17	T: Test to see what happens when attempting to insert a new teacher record using a valid set of data	The record should be added to the database with the details the user has entered. A new TeacherID should be assigned to the new record. A message should display indicating that the record has been added.	As expected. Reference: 19	None Required.
	X: Test what happens when a user does not complete one or more textboxes.	An error message should display saying that “one or more fields have not been completed” and the background colour of any empty fields should become red.	As expected.	None Required.
	X: Test what happens when a user completes previously incomplete textboxes but then clears another textbox that did have information previously entered and attempts to insert	An error message should display saying that “one or more fields have not been completed”. The textboxes that have now been completed should have their backgrounds changed back to white and the textbox that has now been cleared should have its background become red.	As expected.	None Required.
18	T: Test what happens when a user attempts to delete a valid record.	A message should initially display asking if the user really wants to delete the record. I expect the record the user has selected to be deleted from the database and a confirmation message to display.	As expected. Reference: 20	None Required.
	E: Test what happens when a user attempts to delete a record that does not exist. I will clear the surname field on the form to test this.	A message should initially display asking if the user really wants to delete the record. I expect an error message to display indicating that the student record the user	As expected.	None Required.

		has entered does not exist in the database.		
19	T: Test what happens when a user attempts to navigate away from the page but details on the form have not been edited.	I expect the form to close immediately and the user to be re-directed to the form they want to access.	As expected.	None Required.
	E: Test what happens when navigating away from the form but values on the form have been edited.	I expect a confirmation message to display asking the user if they really want to navigate away from the page.	As expected.	None Required.

Main Menu Form

Test No.	Input Data/Action	Expected Outcome	Actual Outcome	Comments and Actions
20	Test what happens when a user clicks a button that does not require an academic year to be selected.	I expect the user to be re-directed to the form that the button the user clicks corresponds to.	As expected.	None Required.
21	T: Test what happens when a user has selected an academic year and clicks a button that requires an academic year to be selected.	I expect the user to be re-directed to the form that the button the user clicks corresponds to.	As expected.	None Required.
	E: Test what happens when a user has not selected an academic year and clicks a button that requires an academic year to be selected.	I do not expect the user to be re-directed to the form that the button the user clicks corresponds to. Instead an error message should display indicating that an academic year needs to be selected.	As expected. Reference: 21	None Required.
22	Test what happens when the user attempts to resize this form using the maximize window option or by dragging the boundaries of the form with the mouse.	Nothing should happen and the form's size should not be altered.	As expected.	None Required.

Student Classes Form

Test No.	Input Data/Action	Expected Outcome	Actual Outcome	Comments and Actions
23	T: Test what happens when a user attempts to filter their Student's selection using a valid name. ("Thomas")	I expect the data grid to return the student records that match the name the user has entered.	As expected.	None Required.
	E: Test what happens when a non-valid name is entered into the student filter selection.	I expect the data grid to not display any student records and for a message box to display asking the user to input a valid student name.	As expected.	None Required.
	X: Test what happens when a user does not type any information into the student filter selection box but still clicks search.	I expect all student rows to be displayed in the data grid.	As expected.	None Required.
24	Test to see what happens when the view classes button that is a part of the view a student's class's grid is clicked.	I expect a list of all the classes the student that has been selected to be returned in the data grid.	As expected. Reference: 22	None Required.
25	T: Test what happens when a user attempts to filter their Classes selection using a valid department ("Science")	I expect all Science classes to be displayed in the data grid.	As expected.	None Required.
	X: Test what happens when a user chooses to filter by department but then does not actually select a department from the drop-down list.	I expect all classes for the academic year the user has selected to be displayed.	As expected.	None Required.
26	Test to see what happens when the view students button that is a part of the "add and remove students from classes" data grid is clicked.	I expect all student records that are a part of the class that a user selects to be returned and displayed in the data grid.	As expected. Reference: 23	None Required.
27	Test to see what happens when the Add students button that is a part of the "add and remove	I expect the add student to class form to be displayed with the class selection grid on the add student to class	As expected. Reference: 24	None Required.

	students from classes” data grid is clicked.	form matching the class the user has clicked.		
28	Test to see what happens when the remove student button that is a part of the “add and remove students from classes” data grid is clicked. This is achieved after viewing the students within a class.	I expect the remove student from class form to display/pop-up and the details of the student the user has selected to remove from the class get passed to this form.	As expected. Reference: 25	None Required.
29	Test what happens when the print icon is clicked by the user	I expect a print dialog box to appear which will allow the user to select where to print a copy of the form and how many copies etc. the form should then be printed by the designated printer in landscape orientation.	As expected.	None Required.
30	Test what happens when a user attempts to return to the main menu and clicks the main menu link label.	The user should be redirected to the main menu.	As expected.	None Required.

Remove Student Class Form

Test No.	Input Data/Action	Expected Outcome	Actual Outcome	Comments and Actions
31	Test what happens when the user clicks the Remove from Class button.	A message should display to the user which asks them to confirm their choice, if the user clicks yes then the student should be removed from the selected class but if they select no nothing should happen.	As expected. Reference: 26	None Required.
32	Test what happens when a user attempts to return to the main menu and clicks the main menu link label.	The user should be redirected to the main menu.	As expected.	None Required.

Add Student Class Form

Test No.	Input Data/Action	Expected Outcome	Actual Outcome	Comments and Actions
33	T: Test what happens when a user attempts to filter their Student's selection using a valid name. ("Thomas")	I expect the data grid to return the student records that match the name the user has entered.	As expected.	None Required.
	T: Test what happens when a user attempts to filter their Student's selection using a valid form group.	I expect the data grid to not display any student records that are a part of the selected form.	As expected.	None Required.
	E: Test what happens when a non-valid name is entered into the student filter selection.	I expect the data grid to not display any student records and for a message box to display asking the user to input a valid student name.	As expected. Reference: 27	None Required.
	X: Test what happens when a user does not type any information into the student filter selection box but still clicks search.	I expect all student rows to be displayed in the data grid.	As expected.	None Required.
34	Test what happens when a user clicks the Select button that is a part of the data grid that appears after filtering students.	I expect the data grid to disappear and a new group box to display which contains some details about the student the user has selected and a new button to appear.	As expected. Reference: 28	None Required.
35	Test what happens when a user clicks the Add selected student button	A message box should display indicating that the student has been successfully added to the current selected class and the record should be added to the dataset. Then the form should close.	As expected. Reference: 29	None Required.
36	Test what happens when a user attempts to add a student to a class that they are already a member of	An error message should display and the student record should not be added to the database as this would cause	As expected. Reference: 30	None Required.

		duplication.		
37	Test what happens a user attempts to search for another student after a student has already been selected.	I expect the selected student group box to disappear and the new filter selection to appear in a data grid.	As expected.	None Required.

Teacher Classes Form

Test No.	Input Data/Action	Expected Outcome	Actual Outcome	Comments and Actions
38	T: Test what happens when a user attempts to filter their Class selection using a valid Class group name. ("39")	I expect the data grid to return the Class records that match the Class the user has entered.	As expected.	None Required.
	E: Test what happens when a non-valid Class name is entered into the Class filter selection.	I expect the data grid to not display any class records.	As expected.	None Required.
	X: Test what happens when a user does not type any information into the Class filter selection box but still clicks search.	I expect all Class rows to be displayed in the data grid.	As expected.	None Required.
39	Test to see what happens when a user clicks any Edit class button which is a part of the data grid is clicked.	I expect the Edit Class form to display/pop-up. I also expect all fields on the edit class form to be automatically filled in with all the details of the class the user has selected to edit.	As expected. Reference: 31	None Required.
40	T: Test to see what happens when attempting to insert a new Class record using a valid set of data	The record should be added to the database with the details the user has entered. A new Class ID should be assigned to the new record. A message should display indicating that the record has been added.	As expected. Reference: 32	None Required.
	E: test to see what happens if a user attempts to create a new class with all data being valid except the teacher first name and surname.	I expect a message box to display indicating that the teacher the user attempted to assign a class to do not exist and that a row is not inserted into the database.	As expected. Reference: 33	None Required.

	A valid teacher that is a part of the system must be selected.			
41	X: test what happens when a user does not complete one or more textboxes.	An error message should display saying that “one or more fields have not been completed” and the background colour of any empty fields should become red.	As expected.	None Required.
	X: test what happens when a user completes previously incomplete textboxes but then clears another textbox that did have information previously entered and attempts to insert	An error message should display saying that “one or more fields have not been completed”. The textboxes that have now been completed should have their backgrounds changed back to white and the textbox that has now been cleared should have its background become red.	As expected.	None Required.
42	Test what happens when the print icon is clicked by the user	I expect a print dialog box to appear which will allow the user to select where to print a copy of the form and how many copies etc. the form should then be printed by the designated printer in landscape orientation.	As expected.	None Required.

Edit Classes Form

Test No.	Input Data/Action	Expected Outcome	Actual Outcome	Comments and Actions
43	Test to see what happens if a user attempts to navigate away from the page but no details have been edited.	The user should be immediately re-directed to their desired navigation choice and the form should close.	As expected.	None Required.
44	Test to see what happens if a user attempts to navigate away from the page but details have been edited.	A message box should appear asking the user if they really want to navigate away from the page. If yes is clicked then the user should be re-directed to their desired navigation choice and the form should close. If no is clicked nothing should happen.	As expected.	None Required.

45	Test what happens when the user attempts clicks the edit button with a valid set of data.	The class the user has chosen to edit should be updated with the new details that have been entered. A message box should display indicating that the record has been updated successfully.	As expected.	None Required.
46	X: test what happens when a user does not complete one or more textboxes.	An error message should display saying that "one or more fields have not been completed" and the background colour of any empty fields should become red.	As expected.	None Required.
	X: test what happens when a user completes previously incomplete textboxes but then clears another textbox that did have information previously entered and attempts to insert	An error message should display saying that "one or more fields have not been completed". The textboxes that have now been completed should have their backgrounds changed back to white and the textbox that has now been cleared should have its background become red.	As expected.	None Required.
47	T: test what happens when a new valid teacher is entered as the teacher of the class.	The class should be edited and the teacher id of the new teacher should be edited as the teacher id of the teacher that teaches the edited class.	As expected.	None Required.
	E: Test what happens if the user clicks edit class but a valid teacher has not been selected.	The class should not be edited and an error message should display indicating that the teacher does not exist.	As expected. Reference:34	None Required.
48	Test what happens when the delete button is clicked and a valid set of class details has been entered.	A dialog box should display that asks if the user really wants to delete the record. If yes, then the record should be deleted. If no then nothing should happen. A message box confirming the deletion should also be displayed.	As expected.	None Required.
49	Test to see what happens if a user attempts to delete a class that does not even exist within the system.	A message box should display indicating that the entered record does not exist.	As expected. Reference: 35	None Required.

My Classes Form

Test No.	Input Data/Action	Expected Outcome	Actual Outcome	Comments and Actions
50	Test what happens when the view student's button that is a part of the data grid is clicked by a potential teacher.	I expect a list of all the students within the class the user has selected to appear in the data grid.	As expected. Reference: 36	None Required.
51	Test what happens when the add student button that is a part of the data grid is clicked by a potential teacher.	I expect the add student class form to display/pop-up and the details of the class the user wants to add a student to passed to the form as the current selected class.	As expected. Reference: 37	None Required.
52	Test what happens when a user clicks the remove student button that is a part of the data grid.	I expect the remove student class form to display/pop-up with the details of the student the user has selected to remove passed to the form and displayed.	As expected. Reference: 38	None Required.
53	Test what happens when the user clicks the clear selection button.	I expect the entire form to reload with the data grid displaying the teacher's classes for the chosen academic year.	As expected. Reference: 39	None Required.
54	Test what happens when the print icon is clicked by the user	I expect a print dialog box to appear which will allow the user to select where to print a copy of the form and how many copies etc. the form should then be printed by the designated printer in landscape orientation.	As expected.	None Required.

My Class Grades Form

Test No.	Input Data/Action	Expected Outcome	Actual Outcome	Comments and Actions
55	Test what happens when the view Grades button that is a part of the data grid is clicked by a potential teacher.	A list of student's should fill the data grids that are a part of the selected class. Their grades that they have achieved for the subject within the class should be displayed along with the details of the qualification.	As expected. Reference: 40	None Required.
56	Test what happens when the Edit Grade button that is a part of the data grid is clicked by a potential teacher.	I expect the edit grade form to display/pop-up and the details of the grade the user wants to edit to be passed to the form as the current selected grade.	As expected. Reference: 41	None Required.
57	Test what happens when the view Grades button that is a part of the data grid is clicked by a potential teacher if they want to view all of a student's grades to see how they are performing elsewhere.	A list of all the grades achieved by the selected student should be displayed in the data grid.	As expected. Reference: 42	None Required.
58	Test what happens when the user clicks the clear selection button.	I expect the entire form to reload with the data grid displaying the teacher's classes for the chosen academic year.	As expected. Reference: 43	None Required.
59	Test what happens when the print icon is clicked by the user	I expect a print dialog box to appear which will allow the user to select where to print a copy of the form and how many copies etc. the form should then be printed by the designated printer in landscape orientation.	As expected.	None Required.

Grade Predictions Form

Test No.	Input Data/Action	Expected Outcome	Actual Outcome	Comments and Actions
60	T: Filter student selection on the form by typing the first name of a student into the value selection box. Using the first name "Will"	I expect the data grid to only return student records where either the first name or surname is similar to "Will"	As expected.	None Required.
	E: Test what happens when an invalid name is entered by the user into the filter selection. Test using the Invalid name "ifhejyp"	I expect the data grid to be filled with zero rows and an error message to display to the user.	As expected.	None Required.
	X: Test what happens when a user does not enter a name into the filter selection but nevertheless clicks the search button.	I expect all student records for the chosen dataset year to be selected in the data grid.	As expected.	None Required.
61	T: Filter student selection by form and display all students from the form 13W.	I expect all students that are in the form group 13W to be displayed in the data grid.	As expected. Reference: 44	None Required.
62	Test what happens when the view Grades button that is a part of the data grid is clicked by a user if they want to view predictions per student.	I expect a list of all the grades for the selected student to populate the data grid.	As expected. Reference: 45	None Required.
63	Test what happens when the predicted grades button that is a part of the data grid is clicked by a user if they want to view predictions per student.	I expect a second data grid to appear with the predicted grades for every possible A2 subject in the system based on the selected student's achieved grades.	As expected. Reference: 46	None Required.
64	T: Test what happens when trying to view an entire form groups predicted grades and the user wants to select a form from the academic year of 2014.	I expect another combo box field to display that only contains form groups that were a part of the 2014 academic year.	As expected.	None Required.
	T: Test what happens when a valid year and form have been selected and the user clicks the search button.	I expect the data grid to be filled with a list of the students within that form and a list of predicted grades for every single student within the form for	As expected. Reference: 47	None Required.

		every possible A2 subject.		
	X: Test what happens if a user changes the academic year selection to blank and attempts to search.	I expect the search button and form combo box field to immediately disappear when the value of the year combo box is blank. They should re-appear when a year is then selected again.	As expected.	None Required.
65	T: Test what happens when trying to view an entire Classes predicted grades and the user wants to select a class from the academic year of 2014 and the subject maths.	I expect another combo box field to display that only contains classes that were a part of the 2014 academic year and were maths classes.	As expected.	None Required.
	X: Test what happens if a user changes the subject selection to blank and attempts to search.	I expect the search button and class combo box field to immediately disappear when the value of the subject combo box is blank. They should re-appear when a subject is then selected again.	As expected.	None Required.
	T: Test what happens when a valid year, class and subject have been selected and the user clicks the search button.	I expect the data grid to be filled with a list of the students within that class and a list of predicted grades for every single student within the class for every possible A2 subject.	As expected.	None Required.
	X: Test what happens if a user changes the academic year selection to blank and attempts to search.	I expect the search button and class combo box field to immediately disappear when the value of the academic year combo box is blank. They should re-appear when an academic year is then selected again.	As expected.	None Required.
66	T: Test what happens when a valid academic year has been selected and the user clicks the search button.	I expect the data grid to be filled with a list of the students within that academic year and a list of predicted grades for every single student within the academic year for every possible A2 subject.	As expected.	None Required.
	X: Test what happens if a user changes the academic year selection to blank and attempts to	I expect the search button and class combo box field to immediately disappear when the value of the	As expected.	None Required.

	search.	academic year combo box is blank. They should re-appear when an academic year is then selected again.		
67	Test what happens when the user clicks the clear selection button.	I expect the entire form to reload with the data grid displaying the teacher's classes for the chosen academic year.	As expected.	None Required.
	Test what happens when the print icon is clicked by the user	I expect a print dialog box to appear which will allow the user to select where to print a copy of the form and how many copies etc. the form should then be printed by the designated printer in landscape orientation.	As expected.	None Required.
68	Test what happens when a user clicks the create spread sheet button which is only visible when a user has generated some predicted grades.	I expect a save file dialog box to display/pop-up which allows a user to select where they would like to save the spread sheet file to. When a user clicks ok a CSV formatted file of the data grid contents should be created in the chosen file path with the name the user has selected. A message indicating the file creation should also appear.	As expected. Reference: 48	None Required.

Grade Form

Test No.	Input Data/Action	Expected Outcome	Actual Outcome	Comments and Actions
69	Test what happens when the user clicks the clear selection button.	I expect the entire form to reload with the data grid displaying the teacher's classes for the chosen academic year.	As expected.	None Required.
70	Test what happens when the print icon is clicked by the user	I expect a print dialog box to appear which will allow the user to select where to print a copy of the form and how many copies etc. the form should then be printed by the designated printer in landscape	As expected.	None Required.

		orientation.		
71	Test what happens when the Edit Grade button that is a part of the data grid is clicked by a potential teacher.	I expect the edit grade form to display/pop-up and the details of the grade the user wants to edit to be passed to the form as the current selected grade.	As expected. Reference: 49	None Required.
72	See what happens when the add grade button is clicked.	I expect the add grade form to open and this form to close.	As expected. Reference: 50	None Required.
73	X: Test what happens when the search button is clicked but no filters have been used	I expect all student grades within the academic year that is selected to be populated in the data grid.	As expected.	None Required.
	T: Test what happens when a user filter's only by the first name 'Will'	All grades that belong to students with a first name that is like 'will' should populate the data grid.	As expected.	None Required.
	T: Test what happens when a user tries to filter by the first name 'will' and the surname which is like 'mur'	I expect all of the grades corresponding to the student named William Muriel to populate the data grid.	As expected.	None Required.
	T: Test what happens when a user tries to filter by the first name 'will' and the surname which is like 'mur' and then by the subject 'ICT'	I expect the data grid to return only one record. The record that shows William Muriel's grade for ICT.	As expected.	None Required.
	E: Test what happens when a user tries to filter by the first name 'will' and the surname which is like 'mur' and then by the subject 'ICT' and then tries to filter by the grade 'C'	I expect zero records to populate the data grid since William Muriel's ICT grade is a 'B'.	As expected.	None Required.
	E: Test what happens when a user tries to filter by the first name 'will' and the surname which is like 'mur' and then by the subject 'ICT' and then tries to filter by the grade 'B' and the key stage 2.	I Expect zero records to populate the data grid since William Muriel's ICT grade is of the level key stage 4 even though the other details are correct.	As expected.	None Required.
	T: Test what happens when a user tries to filter by the first name 'Joe' and the surname which is like 'call' and then by the subject 'French' and then tries to filter by the grade 'C' and the key stage 4.	I expect the data grid to be populated with Joe Callum's French GCSE grade record.	As expected. Reference: 51	None Required.

Add Grade Form

Test No.	Input Data/Action	Expected Outcome	Actual Outcome	Comments and Actions
72	Test what happens when the print icon is clicked by the user	I expect a print dialog box to appear which will allow the user to select where to print a copy of the form and how many copies etc. the form should then be printed by the designated printer in landscape orientation.	As expected.	None Required.
73	T: Filter student selection on the form by typing the first name of a student into the value selection box. Using the first name "Will"	I expect the data grid to only return student records where either the first name or surname is similar to "Will"	As expected.	None Required.
	T: filter student selection by the form group '12P' and then click search.	I expect the data grid to be populated with a list of all the students that are in the form group 12P.	As expected. Reference: 52	None Required.
	E: Test what happens when an invalid name is entered by the user into the filter selection. Test using the Invalid name "ifhejyp"	I expect the data grid to be filled with zero rows and an error message to display to the user.	As expected.	None Required.
	X: Test what happens when a user does not enter a name into the filter selection but nevertheless clicks the search button.	I expect all student records for the chosen dataset year to be selected in the data grid.	As expected.	None Required.
74	Test what happens when the 'view grades' button that is a part of the data grid is clicked.	I expect the data grid to be populated with all grades that the selected student has achieved. I also expect a new group box to appear which allows an admin user to add a new grade for the selected student.	As expected. Reference: 52	None Required.
75	Test what happens when the 'Edit grades' button that is a part of the data grid is clicked.	I expect the edit Grades form to display/pop-up to the user with the details of the grade record being passed to the edit grade form.	As expected. Reference: 53	None Required.

76	Test what happens when a user attempts to insert a new grade into the system but the correct fields have not been completed.	I expect an error message to display indicating that fields have not been completed. I also expect the background colour of any incomplete fields to become red.	As expected. Reference: 54	None Required.
	Test what happens if a user completes a previously incomplete field but then clears a previously complete field.	I expect the field that did have a background colour of red to become white and the background colour of the field that was white to become red.	As expected.	None Required.
77	Test what happens if a user attempts to add a key stage 4 grade with a level value e.g. 5.	I expect an error message to display.	As expected. Reference: 55	None Required.
78	Test what happens if a user attempts to add a new incorrect grade for a non-kS2 subject e.g. a level 3 in ICT.	I expect an error message to display.	As expected.	None Required.
79	Test what happens if a user completes all fields with valid information and clicks the add grade button.	I expect a new grade with a new generated grade ID to be inserted into the system with the corresponding details in the completed fields.	As expected. Reference: 56	None Required.
79 (A)	Test what happens if a user completes all fields with valid information and clicks the add grade button but there is already a grade for that subject.	I expect an error message to display.	As expected. Reference: 57	None Required.
80	Test what happens if a user attempts to navigate from the page and fields have been edited.	I expect a confirmation message box to display asking if the user really wants to navigate away from the page.	As expected.	None Required.
	Test what happens if a user attempts to navigate from the page and fields have not been edited.	I expect the user to be navigated away from the page to their selected navigation choice immediately.	As expected.	None Required.

Edit Grade Form

Test No.	Input Data/Action	Expected Outcome	Actual Outcome	Comments and Actions
81	T: Test what happens if a user clicks the edit grade button with a valid set of completed fields.	I expect the grade to be edited with the new values and for an appropriate message box to display indicating this change.	As expected. Reference: 58	None Required.
	E: Test what happens if a user attempts to edit a key stage 4 grade to a level value e.g. 5.	I expect an error message to display.	As expected. Reference: 59	None Required.
	X: Test what happens if a user clears a selection of any fields and attempts to edit the record.	I expect an error message to display and the background colour of any incomplete fields to turn red.	As expected.	None Required.
	X: Test what happens if a user completes a previously incomplete field but then clears a previously complete field.	I expect the field that did have a background colour of red to become white and the background colour of the field that was white to become red.	As expected.	None Required.
82	Test what happens if a user attempts to navigate from the page and fields have been edited.	I expect a confirmation message box to display asking if the user really wants to navigate away from the page.	As expected.	None Required.
	Test what happens if a user attempts to navigate from the page and fields have not been edited.	I expect the user to be navigated away from the page to their selected navigation choice immediately.	As expected.	None Required.
83	Test what happens when the delete button is clicked and the grade details entered match an existing record.	First I expect a confirmation dialog box to display asking the user if they really want to delete the record. Then, I expect the grade record to be deleted from the database. Finally a message should display confirming the deletion of the record.	As expected. Reference: 60	None Required.
	Test what happens when the delete button is clicked and the grade details entered do not match an existing record.	I expect an error message to display informing the user that the entered record does not exist.	As expected. Reference: 61	None Required.

Developer Features Form

Test No.	Input Data/Action	Expected Outcome	Actual Outcome	Comments and Actions
84	Test what happens when the print icon is clicked by the user	I expect a print dialog box to appear which will allow the user to select where to print a copy of the form and how many copies etc. the form should then be printed by the designated printer in landscape orientation.	As expected.	None Required.
85	Test what happens when the user selects the 'add a new level' admin action.	I expect the add a new level group box to display and the data grid to show all current level types in the system.	As expected. Reference: 62	None Required.
	Test what happens when the user tries to add a new level without completing the field.	The field's background should turn red and an error should display	As expected. Reference: 63	None Required.
	Test what happens when the user types a new level and clicks add.	I expect the new level to be added to the system and the data grid to refresh, showing the new level with a dynamically generated ID.	As expected. Reference: 64	None Required.
	Test what happens when the delete record button is clicked on the data grid	The level that has been selected for deletion should be deleted if the user clicks yes once the confirmation dialog box displays. A deleted message should also display.	As expected. Reference: 65	None Required.
86	Test what happens when the user selects the 'add a new grade type' admin action.	I expect the add a new grade type group box to display and the data grid to show all current grade types in the system.	As expected.	None Required.
	Test what happens when the user tries to add a new grade type without completing the field.	The field's background should turn red and an error should display.	As expected.	None Required.
	Test what happens when the user types a new grade type and clicks add.	I expect the new grade type to be added to the system and the data grid to refresh, showing the new	As expected.	None Required.

		grade type with a dynamically generated ID.		
	Test what happens when the delete record button is clicked on the data grid	The grade type that has been selected for deletion should be deleted if the user clicks yes once the confirmation dialog box displays. A deleted message should also display.	As expected.	None Required.
87	Test what happens when the user selects the 'add a new dataset year' admin action.	I expect the add a new dataset year group box to display and the data grid to show all current dataset year types in the system.	As expected.	None Required.
	Test what happens when the user tries to add a new dataset year without completing the field.	The field's background should turn red and an error should display.	As expected.	None Required.
	Test what happens when the user types a new dataset year and clicks add.	I expect the new dataset year to be added to the system and the data grid to refresh, showing the new dataset year with a dynamically generated ID.	As expected.	None Required.
	Test what happens when the delete record button is clicked on the data grid	The dataset year that has been selected for deletion should be deleted if the user clicks yes once the confirmation dialog box displays. A deleted message should also display.	As expected.	None Required.
88	Test what happens when the user selects the 'add a new subject' admin action.	I expect the add a new subject group box to display and the data grid to show all current subjects in the system.	As expected.	None Required.
	Test what happens when the user tries to add a new subject without completing one or more fields	The fields that have not been completed should have their backgrounds changed to red and an error message should display.	As expected.	None Required.
	Test what happens when the user tries to add a new subject but now completes a previously incomplete field.	The background colour of the field that was previously incomplete should now become white.	As expected.	None Required.
	Test what happens when the user types a new	I expect the new subject to be added to the system	As expected.	None Required.

	subject that is valid and clicks add.	and the data grid to refresh, showing the new subject with a dynamically generated ID.		
	Test what happens when the delete record button is clicked on the data grid	The subject that has been selected for deletion should be deleted if the user clicks yes once the confirmation dialog box displays. A deleted message should also display.	As expected.	None Required.
89	Test what happens when the user selects the 'add a new login' admin action.	I expect the add a new login group box to display and the data grid to show all current user logins in the system.	As expected.	None Required.
	Test what happens when the user tries to add a new grade type without completing one or more field.	The field's backgrounds that have not been completed should turn red and an error should display.	As expected.	None Required.
	Test what happens when the user tries to add a login but now completes a previously incomplete field.	The background colour of the field that was previously incomplete should now become white.	As expected.	None Required.
	Test what happens when the user types a new valid login and clicks add.	I expect the new login to be added to the system and the data grid to refresh, showing the new login with a dynamically generated ID.	As expected.	None Required.
	Test what happens when the delete record button is clicked on the data grid	The login that has been selected for deletion should be deleted if the user clicks yes once the confirmation dialog box displays. A deleted message should also display.	As expected.	None Required.
90	Test what happens when the user selects the 'edit an existing login' admin action.	I expect the data grid to show all current user logins in the system.	As expected.	None Required.
	Test what happens when the edit record button is clicked on the data grid	I expect the edit existing login group box to display with the values of the selected login the user wants to edit passed into the fields within the group	As expected.	None Required.

		box.		
	Test what happens when the user tries to edit a login without completing one or more field.	The field's backgrounds that have not been completed should turn red and an error should display.	As expected.	None Required.
	Test what happens when the user tries to edit a login but now completes a previously incomplete field.	The background colour of the field that was previously incomplete should now become white.	As expected.	None Required.
	Test what happens when the user edits a new login and clicks add with all fields complete	I expect the new login to be added to the system and the data grid to refresh, showing the new login with a dynamically generated ID. An updated message should also display.	As expected.	None Required.

Grade Percentages Form

Test No.	Input Data/Action	Expected Outcome	Actual Outcome	Comments and Actions
91	Test what happens when the print icon is clicked by the user	I expect a print dialog box to appear which will allow the user to select where to print a copy of the form and how many copies etc. the form should then be printed by the designated printer in landscape orientation.	As expected.	None Required.
92	Test what happens when the user clicks the clear chart selection button.	I expect the entire form to reload with the data grid displaying the teacher's classes for the chosen academic year.	As expected. Reference: 66	None Required.
93	Test what happens when the user clicks the view graph button.	I expect all raw data tables to be non-visible and the graph to become visible.	As expected. Reference: 67	None Required.
94	Test what happens when the user clicks the 'view number of students achieving grade' button.	I expect all raw data tables to become visible and the graph to be non-visible.	As expected. Reference: 68	None Required.

95	Test what happens when the ‘standard grade percentage graph’ option is used as the graph type.	I expect a linear bar chart to represent the data.	As expected. Reference: 69	None Required.
	Test what happens when the “cumulative grade percentage graph” option is used as the graph type.	I expect a cumulative bar chart to represent the data.	As expected. Reference: 70	None Required.
96	Test what happens If the user selects KS4 as the level.	I expect the grade type field to be enabled because predicted grades can be viewed based on KS4 grades.	As expected. Reference: 71	None Required.
	Test what happens If the user selects KS2 as the level.	I expect the grade type field to not be enabled because predicted grades can only be viewed based on KS4 grades.	As expected. Reference: 72	None Required.
97	Test what happens if a user selects the “A2 predicted” grade type.	Predicted grades for that subject should be generated and represented by the graph and raw data tables.	As expected. Reference: 73	None Required.
	Test what happens if a user selects the “Achieved” grade type.	Achieved KS4 or KS2 for that subject should be generated and represented by the graph and raw data tables.	As expected.	None Required.
98	Test what happens if a user selects KS2 but then tries to select a non-key stage 2 subject e.g. business studies.	An error message should display.	As expected. Reference: 74	None Required.
99	Test what happens when the user tries to view statistics without completing one or more fields	The fields that have not been completed should have their backgrounds changed to red and an error message should display.	As expected.	None Required.
	Test what happens when the user tries view statistics but now completes a previously incomplete field.	The background colour of the field that was previously incomplete should now become white.	As expected.	None Required.

Compare Subject Percentages Form

Test No.	Input Data/Action	Expected Outcome	Actual Outcome	Comments and Actions
100	Test what happens when the print icon is clicked by the user	I expect a print dialog box to appear which will allow the user to select where to print a copy of the form and how many copies etc. the form should then be printed by the designated printer in landscape orientation.	As expected.	None Required.
101	Test what happens when the user clicks the clear chart selection button.	I expect the entire form to reload with the data grid displaying the teacher's classes for the chosen academic year.	As expected. Reference: 75	None Required.
102	Test what happens when the user clicks the view graph button.	I expect all raw data tables to be non-visible and the graph to become visible.	As expected. Reference: 76	None Required.
103	Test what happens when the user clicks the 'view number of students achieving grade' button.	I expect all raw data tables to become visible and the graph to be non-visible.	As expected. Reference: 77	None Required.
104	Test what happens when the 'standard grade percentage graph' option is used as the graph type.	I expect a linear line graph to represent the data and a chart legend to be added.	As expected. Reference: 78	None Required.
	Test what happens when the "cumulative grade percentage graph" option is used as the graph type.	I expect a cumulative line graph to represent the data and a chart legend to be added.	As expected. Reference: 79	None Required.
105	Test what happens when the user clicks the toggle chart labels button.	I expect the labels on the chart to either be visible or non-visible.	As expected. Reference: 80	None Required.
106	Test what happens If the user selects KS4 as the level.	I expect the grade type field to be enabled because predicted grades can be viewed based on KS4 grades.	As expected.	None Required.
107	Test what happens if a user selects the "A2 predicted" grade type.	Predicted grades for that subject should be generated and represented by the graph and raw data tables.	As expected. Reference: 81	None Required.

	Test what happens if a user selects the “Achieved” grade type.	Achieved KS4 or KS2 for that subject should be generated and represented by the graph and raw data tables.	As expected. Reference: 82	None Required.
108	Test what happens if a user selects KS2 but then tries to select a non-key stage 2 subject e.g. business studies.	An error message should display.	As expected. Reference: 83	None Required.
109	Test what happens when the user tries to view statistics without completing one or more fields	The fields that have not been completed should have their backgrounds changed to red and an error message should display.	As expected.	None Required.
	Test what happens when the user tries view statistics but now completes a previously incomplete field.	The background colour of the field that was previously incomplete should now become white.	As expected.	None Required.
110	Test what happens after the user has added one series.	The text of the button ‘display graph’ should change to ‘add series’. The graph type and grade type fields should not be enabled anymore.	As expected. Reference: 84	None Required.
111	Test what happens if more than one series is added.	The next series should be added to the graph and represented in a separate raw data table.	As expected. Reference: 85	None Required.
	Test what happens if more than 4 series is added.	An error message should display as a maximum of 4 series can be displayed.	As expected. Reference: 86	None Required.

Compare Department Percentages Form

Test No.	Input Data/Action	Expected Outcome	Actual Outcome	Comments and Actions
112	Test what happens when the print icon is clicked by the user	I expect a print dialog box to appear which will allow the user to select where to print a copy of the form and how many copies etc. the form should then be printed by the designated printer in landscape orientation.	As expected.	None Required.
113	Test what happens when the user clicks the clear chart selection button.	I expect the entire form to reload with the data grid displaying the teacher's classes for the chosen academic year.	As expected.	None Required.
114	Test what happens when the user clicks the view graph button.	I expect all raw data tables to be non-visible and the graph to become visible.	As expected.	None Required.
115	Test what happens when the user clicks the 'view number of students achieving grade' button.	I expect all raw data tables to become visible and the graph to be non-visible.	As expected.	None Required.
116	Test what happens when the 'standard grade percentage graph' option is used as the graph type.	I expect a linear line graph to represent the data and a chart legend to be added.	As expected.	None Required.
	Test what happens when the "cumulative grade percentage graph" option is used as the graph type.	I expect a cumulative line graph to represent the data and a chart legend to be added.	As expected.	None Required.
117	Test what happens when the user clicks the toggle chart labels button.	I expect the labels on the chart to either be visible or non-visible.	As expected.	None Required.
118	Test what happens If the user selects KS4 as the level.	I expect the grade type field to be enabled because predicted grades can be viewed based on KS4 grades.	As expected.	None Required.
119	Test what happens if a user selects the "A2 predicted" grade type.	Predicted grades for that department should be generated and represented by the graph and raw data tables.	As expected. Reference: 87	None Required.
	Test what happens if a user selects the "Achieved" grade type.	Achieved KS4 or KS2 grades for that department should be generated and	As expected. Reference: 88	None Required.

		represented by the graph and raw data tables.		
120	Test what happens if a user selects KS2 but then tries to select a non-key stage 2 department e.g. physical.	An error message should display.	As expected.	None Required.
121	Test what happens when the user tries to view statistics without completing one or more fields	The fields that have not been completed should have their backgrounds changed to red and an error message should display.	As expected.	None Required.
	Test what happens when the user tries view statistics but now completes a previously incomplete field.	The background colour of the field that was previously incomplete should now become white.	As expected.	None Required.
122	Test what happens after the user has added one series.	The text of the button 'display graph' should change to 'add series'. The graph type and grade type fields should not be enabled anymore.	As expected.	None Required.
123	Test what happens if more than one series is added.	The next series should be added to the graph and represented in a separate raw data table.	As expected.	None Required.
124	Test what happens if more than 4 series is added.	An error message should display as a maximum of 4 series can be displayed.	As expected.	None Required.

Compare Class Percentages Form

Test No.	Input Data/Action	Expected Outcome	Actual Outcome	Comments and Actions
125	Test what happens when the print icon is clicked by the user	I expect a print dialog box to appear which will allow the user to select where to print a copy of the form and how many copies etc. the form should then be printed by the designated printer in landscape orientation.	As expected.	None Required.
126	Test what happens when the user clicks the clear chart selection button.	I expect the entire form to reload with the data grid displaying the teacher's classes for the chosen academic year.	As expected.	None Required.
127	Test what happens when the user clicks the view graph button.	I expect all raw data tables to be non-visible and the graph to become visible.	As expected.	None Required.
128	Test what happens when the user clicks the 'view number of students achieving grade' button.	I expect all raw data tables to become visible and the graph to be non-visible.	As expected.	None Required.
129	Test what happens when the 'standard grade percentage graph' option is used as the graph type.	I expect a linear line graph to represent the data and a chart legend to be added.	As expected.	None Required.
	Test what happens when the "cumulative grade percentage graph" option is used as the graph type.	I expect a cumulative line graph to represent the data and a chart legend to be added.	As expected.	None Required.
130	Test what happens when the user clicks the toggle chart labels button.	I expect the labels on the chart to either be visible or non-visible.	As expected.	None Required.
131	Test what happens if the user selects KS4 as the level.	I expect the grade type field to be enabled because predicted grades can be viewed based on KS4 grades.	As expected.	None Required.
132	Test what happens if a user selects the "A2 predicted" grade type.	Predicted grades for that class should be generated and represented by the graph and raw data tables.	As expected. Reference: 89	None Required.
	Test what happens if a user selects the "Achieved" grade type.	Achieved KS4 or KS2 for that class should be generated and represented by the graph and raw data tables.	As expected. Reference: 90	None Required.

133	Test what happens if a user selects KS2 but then tries to select a non-key stage 2 subject e.g. physical.	An error message should display.	As expected.	None Required.
134	Test what happens when the user tries to view statistics without completing one or more fields	The fields that have not been completed should have their backgrounds changed to red and an error message should display.	As expected.	None Required.
	Test what happens when the user tries view statistics but now completes a previously incomplete field.	The background colour of the field that was previously incomplete should now become white.	As expected.	None Required.
135	Test what happens after the user has added one series.	The text of the button 'display graph' should change to 'add series'. The graph type and grade type fields should not be enabled anymore.	As expected.	None Required.
136	Test what happens if more than one series is added.	The next series should be added to the graph and represented in a separate raw data table.	As expected.	None Required.
137	Test what happens if more than 4 series is added.	An error message should display as a maximum of 4 series can be displayed.	As expected.	None Required.
138	Test what happens when the first three fields (subject, year and level) have been completed.	A fourth field should generate with all the classes that match the selected criteria.	As expected. Reference: 91	None Required.

Navigation Testing

The table on the following page tests the unidirectional navigation links between each from. Since the test plan form my navigation I have added a new navigation type to the key to more accurately how links are formed. The orange filled cell indicates that the form that was used to navigate from remains open. The table on the next page uses the key below:

Successful Navigation	
Navigates to form in a separate instance (pop/up)	
Unsuccessful Navigation	
No need for Navigation	
Forms are not Linked	

- A. Login.vb
- B. SplashScreen.vb
- C. Main_Menu.vb
- D. MyClasses.vb
- E. MyClassGrades.vb
- F. RemoveStudentClass.vb
- G. StudentClasses.vb
- H. Student.vb
- I. Teacher_Classes.vb
- J. TeacherView.vb
- K. Grade.vb
- L. Grade_Predictions.vb
- M. GradePercentages.vb
- N. EditAdd_Teacher.vb
- O. EditAdd_Student.vb
- P. Edit_Grade.vb
- Q. Edit_Classes.vb
- R. Add_Grade.vb
- S. AddStudentClass.vb
- T. DeveloperFeatures.vb
- U. CompareSubjectPercentages.vb
- V. CompareDepartmentPercentages.vb
- W. CompareClassPercentages.vb

Navigating From: →	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Navigating To: ↓	A	✓	✓																				
A																							
B																							
C	✓				✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
D			✓																				
E			✓																				
F				✓				✓															
G		✓																					
H		✓															✓						
I		✓																	✓				
J			✓															✓					
K			✓																		✓		
L			✓																				
M			✓																				
N													✓										
O													✓										
P					✓										✓						✓		
Q														✓									
R																✓							
S						✓							✓										
T			✓												✓								
U			✓																				
V			✓																				
W			✓																				

Secondary Interview with Robert Nicoll – The Primary Client

During Development of my project I conducted a secondary interview with one of my end-users in order to gain some valued feedback on areas I could improve along with recommendations. It was also important for me to gather an end-users feedback on my chosen design choices.

What is your opinion on me using the SQL server as a system to store the data?

RN: I think this is a great idea, it will allow for safe storage of data within the system and will also allow for data to be validated upon insert and editing.

What do you think about distributing the system using the internal teacher network?

RN: In this modern age of technology using devices such as CD-ROM have become obsolete so I think it is great to be able to distribute software without the need for a physical device to be given to a user.

What do you think about the model I have implemented to predict grades?

RN: The model that you have designed is extremely clever, it makes predicting grades even more accurate than most normal systems available. Most systems only apply a weighting on the A2 side of the prediction but since you are going to apply a weighting on the GCSE side of the prediction it will allow for a more genuine prediction based on the student's achieved grades and the subjects they achieved them in.

Are there any issues you have found with the implemented solution so far, bearing in mind it is not completely finished?

RN: There are not any major issues within the system but some small user problems I have identified include:

-Currently users can type in a data grid, would it be possible to make the data grids in the system read-only?

-Your validation of combo boxes or textboxes not being completed by the user is good, and I like how the background colour changes to red if information has not been entered. However, when information is re-entered the background colour still remains red.

-At the moment if I write information into a data input form and then click to navigate away, it will do so immediately, is there any way you could validate this and make sure the user really wants to navigate away from the page?

-Another minor issue I identified is that currently it is possible for a user to enter multiple grades for the same subject of a single student within the system, is there any way to prevent this?

-I have also identified another data input fault; currently it is possible for a user to enter a levelled grade for a KS4 subject and a non-levelled grade for a KS2 subject.

Signed: Robert Nicoll

Testing Evaluation

The testing I have conducted which has followed my testing plan from my initial solution Design section has allowed me to extensively test the functionality of my system. I have carried out tests for each individual form within my solution which require testing and then have tested individual features on each form. Within my testing I showed that my system is capable of responding to errors appropriately using various forms of validation. My system has been able to manage erroneous errors and extreme errors which I managed to test using boundary data values. Before carrying out each test I have described my expectations of carrying out each test. Furthermore, I have described the tests that I will be doing and where applicable included data that I will test. In order to test the navigation of my system I tested every single navigation link within my system, including button links to test whether the links correctly navigated the user to the correct form. I used the same method of navigation testing which I outlined in my Design but I also added some different Key examples to more accurately describe the results of navigation links.

During the development of my system I also held an interview with one of my primary users, Mr Nicoll. This allowed me to identify problems with my system that I had not thought about and then make the necessary corrections. For this reason I have included this interview within my testing section although it is also discusses some of my design and implementation choices.

I believe my System's testing was carried out very effectively overall and I have managed to show evidence of extensive testing with references per test case. I have not found any major problems within my current system which was unexpected, but there will most likely be improvements I can make for my system on the whole, these are features I will discuss in my Appraisal section.

All other tests within the system have been validated by either Mr Hewitt or Mr Nicoll, who are both primary users of the system.

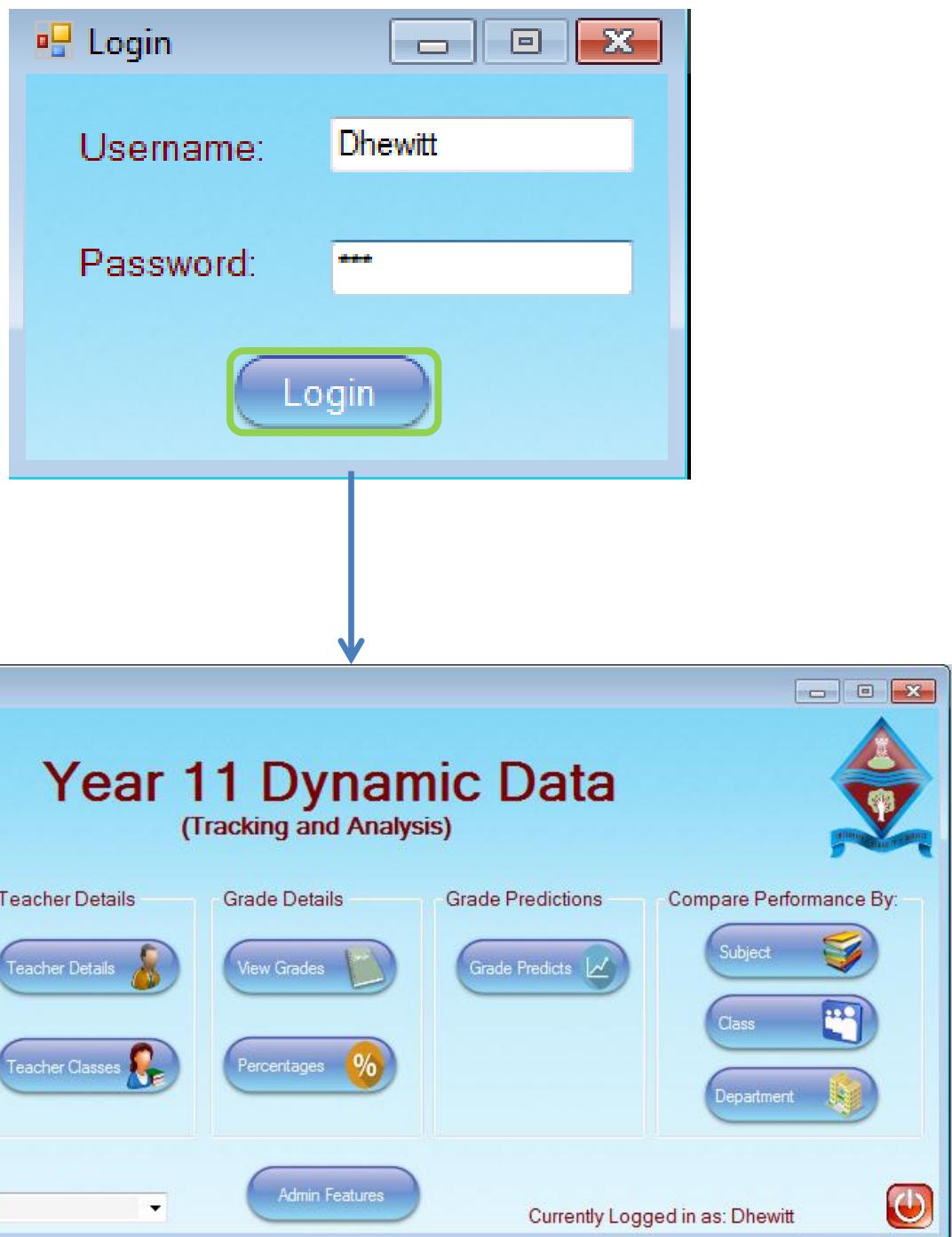


Testing Evidence

All of my testing evidence will be referenced within my Actual Outcome columns of the testing tables, a unique test evidence reference number will be given to each test. Where possible I will also include detailed annotation of what the test evidence illustrates. I have chosen not to use the table I designed in my design section because it was not a practical way of showing screenshots, since I will be attempting to use a maximum of two screenshots per page where possible for my test evidence.

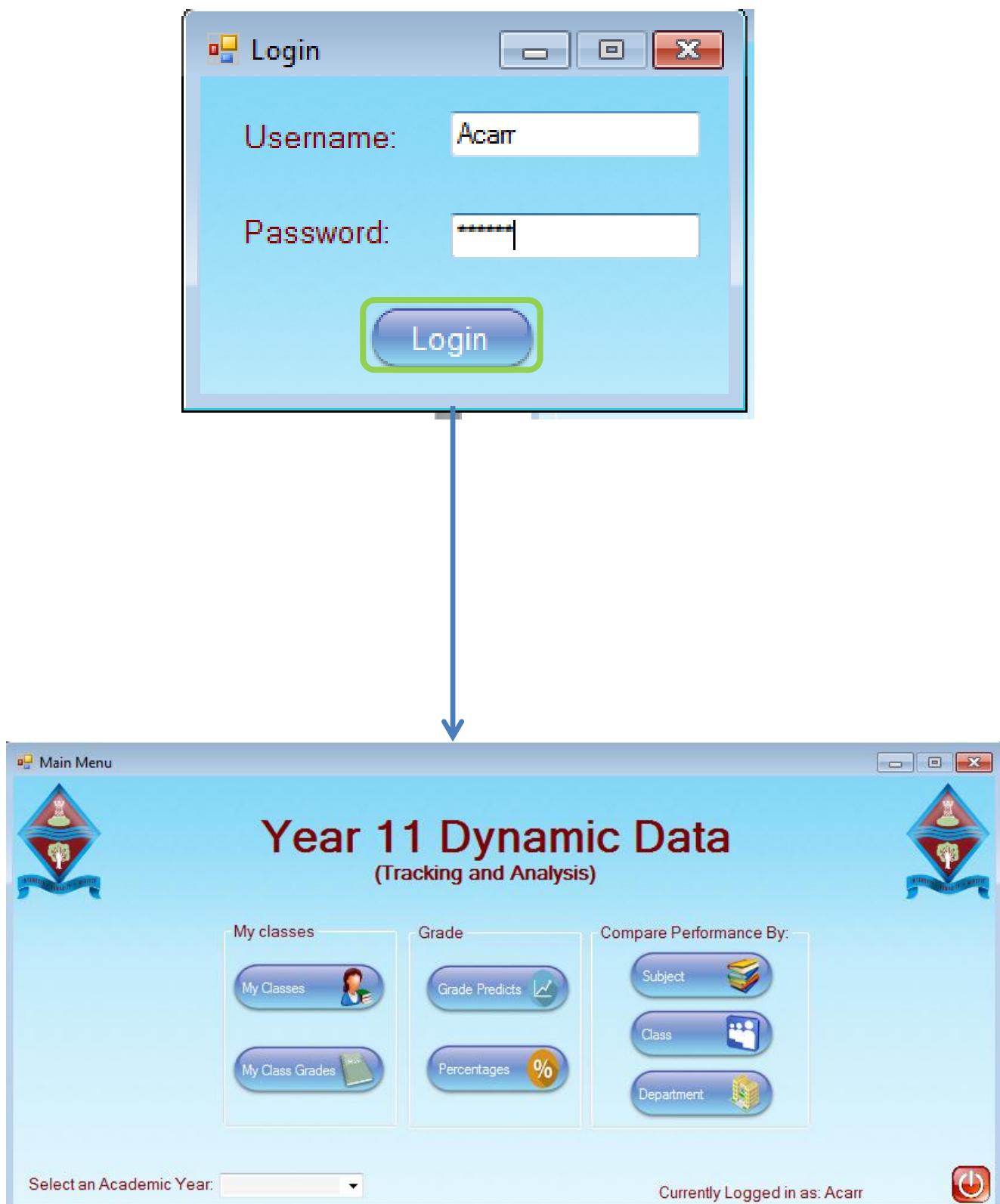
Reference 1

This evidence shows logging into the system as an admin user.



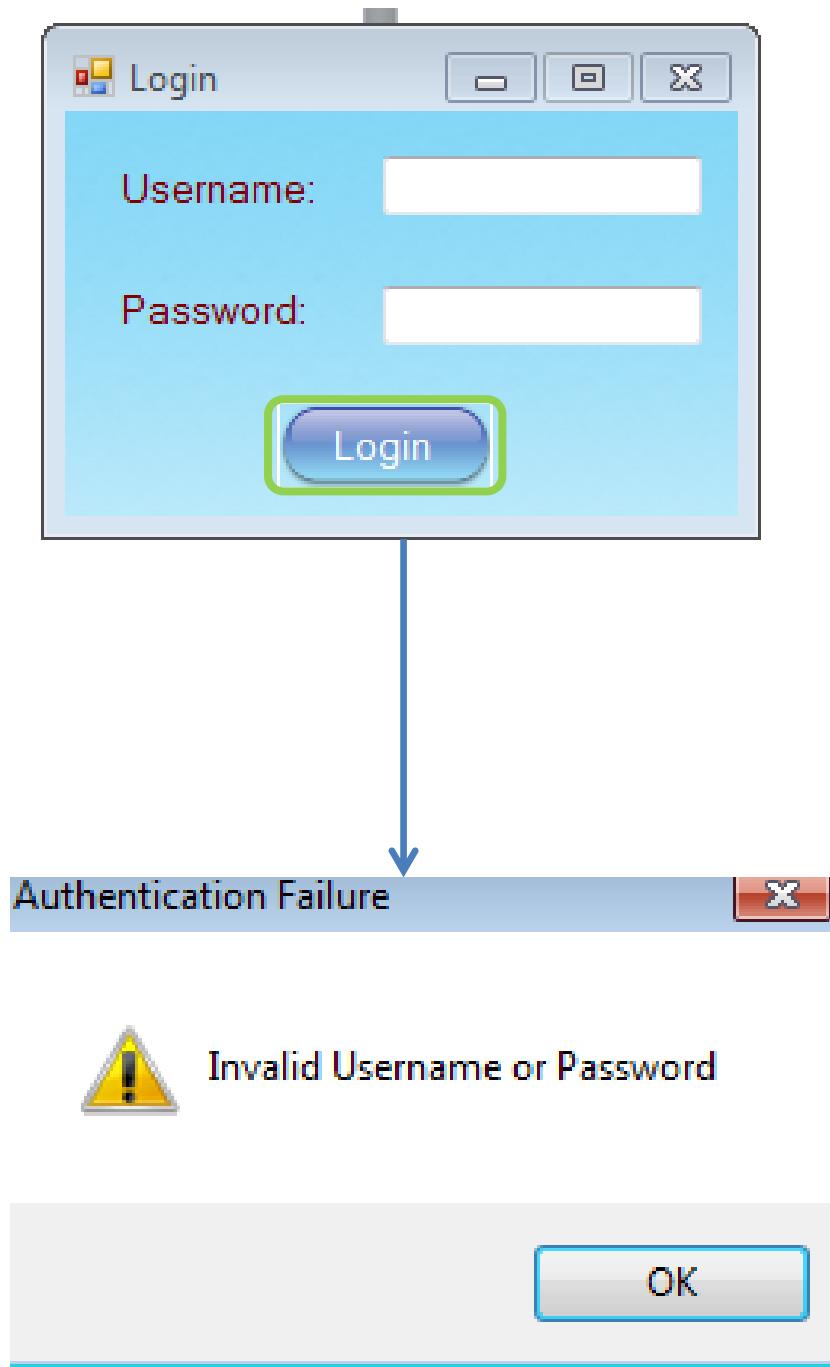
Reference 1 (continued)

This evidence shows logging into the system as a non-admin user.



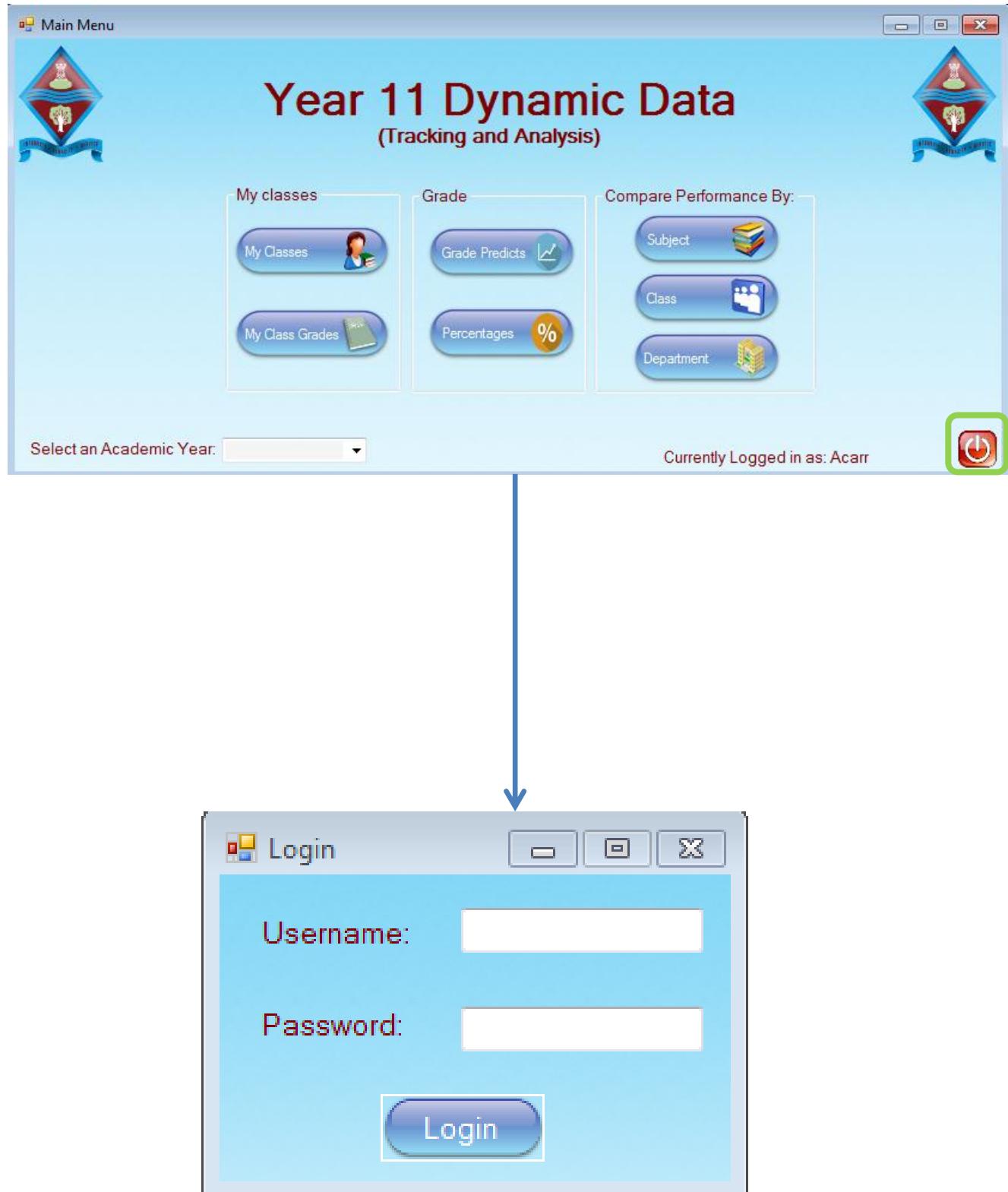
Reference 2

This test is evidence of using boundary data and shows what happens when a user attempts to log in to the system without completing the required text fields.



Reference 3

This test shows what happens when a user logs out from the main menu form.



Reference 4

This test shows what happens when the user attempts to filter students using a name.

Screenshot of the 'Search Student Details' interface showing the search process:

The 'Name' field contains 'will'. A green oval highlights the 'Search' button, indicating it has been clicked.

Below the search results table, a large blue arrow points downwards, indicating the transition to the results page.

Screenshot of the 'Search Student Details' interface showing the search results:

The 'Name' field contains 'will'. The search results table displays four student records:

Firstname	Surname	Gender	FSM	SNS	Ethnicity	Attendance	DateOfBirth	Form	Year	Student Details
William	ABERCROMBIE	M	N	N	White - British	85	31/01/1997	13R	13	<button>Edit Details</button>
William	AKRAM	M	N	N	White and Asian	83.3	11/02/1998	13M	13	<button>Edit Details</button>
William	ALLISON	M	N	N	White - British	100	24/05/1998	13B	13	<button>Edit Details</button>
Will	ALTOFT	M	N	N	White - British	98.3	13/09/1997	13C	13	<button>Edit Details</button>

Reference 5

This evidence shows the result of filtering with an invalid filter selection.

The screenshot shows two instances of the 'Student' application window. The top window is titled 'Search Student Details'. It has a search panel on the left with a dropdown 'Select an attribute:' set to 'Name' and an input field containing 'ifhejyp'. A green oval highlights the 'Search' button. Below this is an 'Add New Student' button. To the right is an 'Import CSV' section with a 'Select an Academic Year:' dropdown and an 'Import CSV file' button. A large blue arrow points downwards from the top window to the bottom window. The bottom window is also titled 'Search Student Details' and has the same layout. However, a modal dialog box titled 'Year 11 Dynamic Grade data' is displayed in the center of the main window area. The dialog contains the message 'Please enter a valid name' and an 'OK' button.

Reference 6

Shows what happens when a user clicks one of the edit details button on the data grid.

The diagram illustrates a user interaction flow between two windows:

- Search Student Details Window:**
 - Header: Main Menu > Student
 - Section: Search Student Details
 - Form dropdown: Form (selected value: 13A)
 - Search button
 - Add New Student button
 - Import CSV section
 - Add a new Dataset Year button
 - Select an Academic Year dropdown
 - Import CSV file button
 - Data Grid (15 rows):

Firstname	Surname	Gender	FSM	SNS	Ethnicity	Attendance	DateOfBirth	Form	Year	Student Details
Sophie	BOND	F	N	N	White - British	100	07/09/1997	13A	13	Edit Details
Rachel	CANNON	F	N	N	White - British	93.3	28/04/1998	13A	13	Edit Details
Tom	CLOUGH	M	N	N	White - British	100	07/03/1998	13A	13	Edit Details
Natalie	COMERY	F	N	N	White - British	91.7	22/03/1998	13A	13	Edit Details (highlighted with a green box)
Max	DAVIES	M	N	N	White - British	96.7	14/09/1997	13A	13	Edit Details
Luke	EYRE	M	N	N	White - British	100	24/09/1997	13A	13	Edit Details
Liam	FOULDS	M	N	N	White - British	96.7	15/11/1997	13A	13	Edit Details
Katy	GRIFFITHS	F	N	N	White - British	100	05/10/1997	13A	13	Edit Details
Jasmine	HOWE	F	N	K	White - British	93.3	12/10/1997	13A	13	Edit Details
James	JOHNSON	M	N	N	White - British	98.3	22/06/1998	13A	13	Edit Details
Jack	KINGSCOTT	M	N	N	White - British	96.7	18/03/1998	13A	13	Edit Details
Jack	LAMB	M	N	N	White - British	95	02/03/1998	13A	13	Edit Details
Emily	PAISH	F	N	N	White - British	100	01/08/1998	13A	13	Edit Details
Elizabeth	PRITCHARD	F	N	N	White - British	98.3	23/05/1998	13A	13	Edit Details
Daniel	SADULA	M	N	N	White - British	93.3	06/03/1998	13A	13	Edit Details
Chloe	SLACK	F	N	N	White - British	100	11/06/1998	13A	13	Edit Details
Catherine	SUTCLIFFE	F	N	N	White - British	96.7	23/06/1998	13A	13	Edit Details

Edit Student Details

Firstname:	Natalie
Surname:	COMERY
Gender:	F
Ethnicity:	White - British
Attendance(%):	91.7
Date Of Birth:	22 March 1998
Special Needs Status(SNS):	N
Entitled To Free School Meals(FSM):	N
Form:	13A
Current School Year:	13

Buttons: Edit, Delete

Reference 7

Shows what happens when the add new student button is clicked. One thing I noticed was that the date of birth field is automatically completed with the current day (the day in which I did this test).

Firstname	Surname	Gender	FSM	SNS	Ethnicity	Attendance	DateOfBirth	Form	Year	Student Details
Sophie	BOND	F	N	N	White - British	100	07/09/1997	13A	13	<button>Edit Details</button>
Rachel	CANNON	F	N	N	White - British	93.3	28/04/1998	13A	13	<button>Edit Details</button>
Tom	CLOUGH	M	N	N	White - British	100	07/03/1998	13A	13	<button>Edit Details</button>
Natalie	COMERY	F	N	N	White - British	91.7	22/03/1998	13A	13	<button>Edit Details</button>
Max	DAVIES	M	N	N	White - British	96.7	14/09/1997	13A	13	<button>Edit Details</button>
Luke	EYRE	M	N	N	White - British	100	24/09/1997	13A	13	<button>Edit Details</button>
Liam	FOULDS	M	N	N	White - British	96.7	15/11/1997	13A	13	<button>Edit Details</button>
Katy	GRIFFITHS	F	N	N	White - British	100	05/10/1997	13A	13	<button>Edit Details</button>
Jasmine	HOWE	F	N	K	White - British	93.3	12/10/1997	13A	13	<button>Edit Details</button>
James	JOHNSON	M	N	N	White - British	98.3	22/06/1998	13A	13	<button>Edit Details</button>
Jack	KINGSCOTT	M	N	N	White - British	96.7	18/03/1998	13A	13	<button>Edit Details</button>
Jack	LAMB	M	N	N	White - British	95	02/03/1998	13A	13	<button>Edit Details</button>
Emily	PAISH	F	N	N	White - British	100	01/08/1998	13A	13	<button>Edit Details</button>
Elizabeth	PRITCHARD	F	N	N	White - British	98.3	23/05/1998	13A	13	<button>Edit Details</button>
Daniel	SADULA	M	N	N	White - British	93.3	06/03/1998	13A	13	<button>Edit Details</button>
Chloe	SLACK	F	N	N	White - British	100	11/06/1998	13A	13	<button>Edit Details</button>
Catherine	SUTCLIFFE	F	N	N	White - British	96.7	23/06/1998	13A	13	<button>Edit Details</button>

Main Menu > Student > Add Student

Add Student Details

Firstname:

Surname:

Gender:

Ethnicity:

Attendance(%):

Date Of Birth:

Special Needs Status(SNS):

Entitled To Free School Meals(FSM):

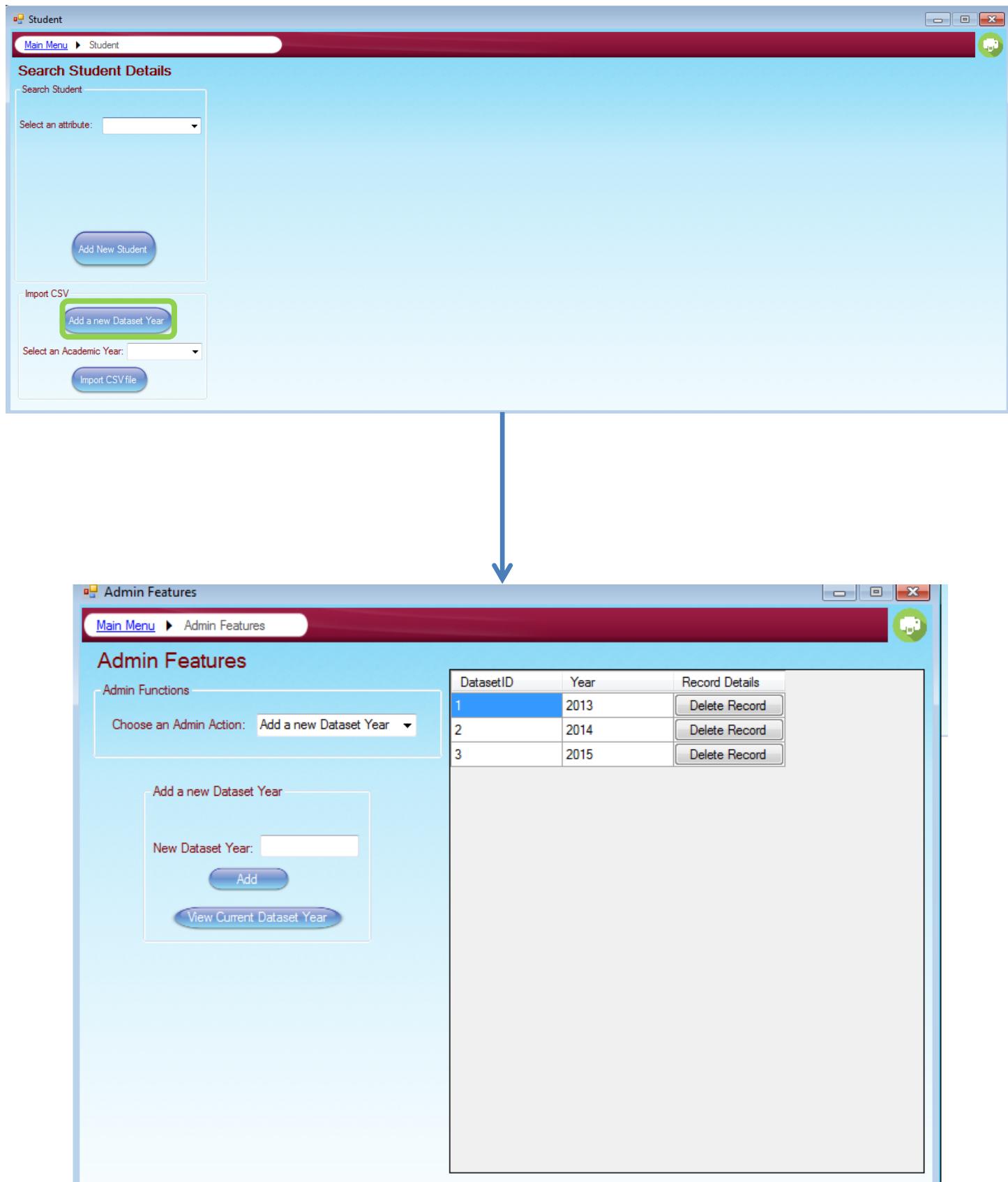
Form:

Current School Year:

Exams Year:

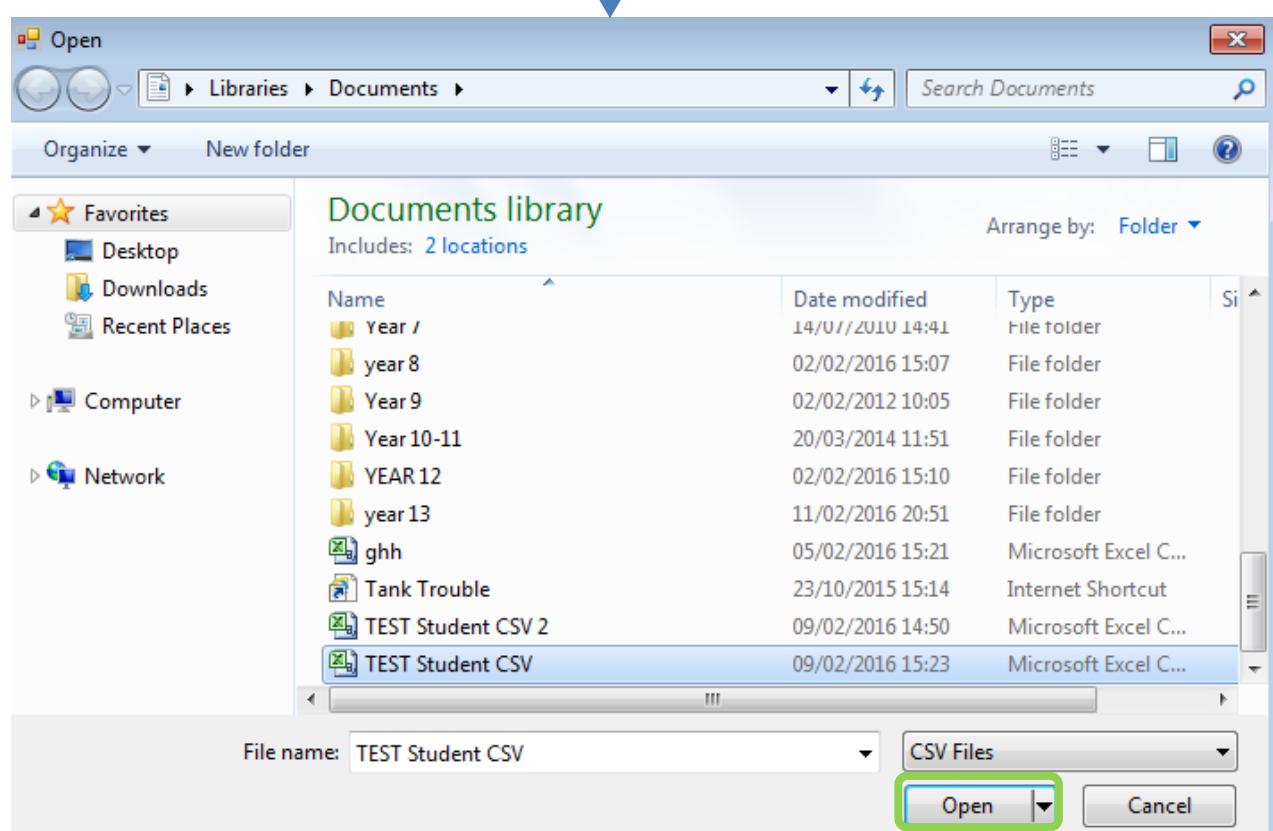
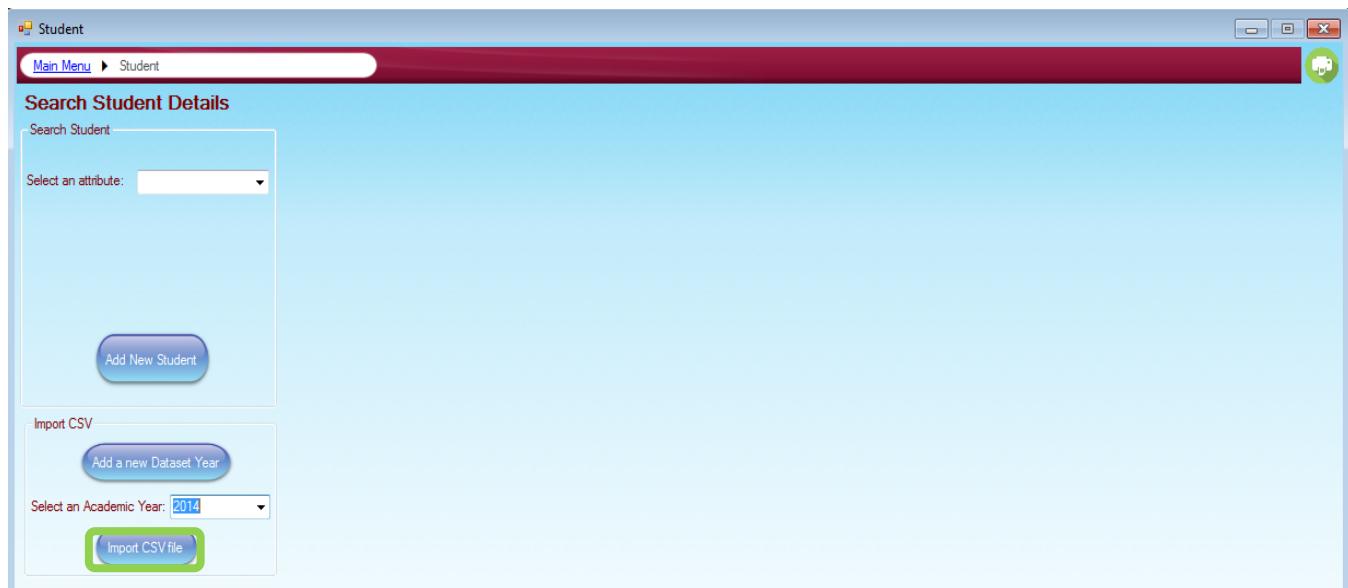
Reference 8

This evidence shows what happens when the user clicks the add a new dataset year button.



Reference 9

This test shows the evidence of importing a large volume of data into the database; in this case student details are imported. The rows that have been inserted are then shown in the data grid with a true or false state depending on whether that record has been successfully inserted.



Data Upload

Insert Data Successful

OK

Student

Main Menu > Student

Search Student Details

Search Student

Select an attribute:

Add New Student

Import CSV

Add a new Dataset Year

Select an Academic Year: 2014

Import CSV file

Firstname	Surname	Gender	Entitled to free schoolmeals	Special Needs Status	Ethnicity	Attendance	Date Of Birth	Form	Year	Inserted (True/False)
TESTNAME	TESTNAME	N	N	N	ANY	1000	23/06/2000	13	19	True
TESTNAME	TESTNAME	N	N	N	ANY	1001	24/06/2000	13	20	True
TESTNAME	TESTNAME	N	N	N	ANY	1002	25/06/2000	13	21	True
TESTNAME	TESTNAME	N	N	N	ANY	1003	26/06/2000	13	22	True
TESTNAME	TESTNAME	N	N	N	ANY	1004	27/06/2000	13	23	True
TESTNAME	TESTNAME	N	N	N	ANY	1005	28/06/2000	13	24	True
TESTNAME	TESTNAME	N	N	N	ANY	1006	29/06/2000	13	25	True
TESTNAME	TESTNAME	N	N	N	ANY	1007	30/06/2000	13	26	True
TESTNAME	TESTNAME	N	N	N	ANY	1008	01/07/2000	13	27	True
TESTNAME	TESTNAME	N	N	N	ANY	1009	02/07/2000	13	28	True
TESTNAME	TESTNAME	N	N	N	ANY	1010	03/07/2000	13	29	True
TESTNAME	TESTNAME	N	N	N	ANY	1011	04/07/2000	13	30	True
TESTNAME	TESTNAME	N	N	N	ANY	1012	05/07/2000	13	31	True
TESTNAME	TESTNAME	N	N	N	ANY	1013	06/07/2000	13	32	True
TESTNAME	TESTNAME	N	N	N	ANY	1014	07/07/2000	13	33	True
TESTNAME	TESTNAME	N	N	N	ANY	1015	08/07/2000	13	34	True

Evidence of data inserted in the actual SQL server database.

Microsoft SQL Server Management Studio

Object Explorer

SQLQuery27.sql...ingleton (67)

```
FROM [dn.ingleton].[dbo].[Student]
```

Results

StudentID	Firstname	Surname	Gender	FSM	SNS	Ethnicity	Attendance	DateOfBirth	Form	Year	DatasetID
562	TESTN...	TESTNAME	N	N	N	ANY	White - B...	2000-06-28	13	24	2
563	TESTN...	TESTNAME	N	N	N	ANY	White - B...	2000-06-29	13	25	2
564	TESTN...	TESTNAME	N	N	N	ANY	Any other...	2000-06-30	13	26	2
565	TESTN...	TESTNAME	N	N	N	ANY	White - B...	2000-07-01	13	27	2
566	TESTN...	TESTNAME	N	N	N	ANY	Any other...	2000-07-02	13	28	2
567	TESTN...	TESTNAME	N	N	N	ANY	White - B...	2000-07-03	13	29	2
568	TESTN...	TESTNAME	N	N	N	ANY	White - B...	2000-07-04	13	30	2
569	TESTN...	TESTNAME	N	N	N	ANY	White - B...	2000-07-05	13	31	2
570	TESTN...	TESTNAME	N	N	N	ANY	White - B...	2000-07-06	13	32	2
571	TESTN...	TESTNAME	N	N	N	ANY	White - B...	2000-07-07	13	33	2
572	TESTN...	TESTNAME	N	N	N	ANY	White - B...	2000-07-08	13	34	2
573	TESTN...	TESTNAME	N	N	N	ANY	White - B...	2000-07-09	13	35	2
574	TESTN...	TESTNAME	N	N	N	ANY	White - B...	2000-07-10	13	36	2
575	TESTN...	TESTNAME	N	N	N	ANY	White - B...	2000-07-11	13	37	2
576	TESTN...	TESTNAME	N	N	N	ANY	Refused	2000-07-12	13	38	2
577	TESTN...	TESTNAME	N	N	N	ANY	White - B...	2000-07-13	13	39	2
578	TESTN...	TESTNAME	N	N	N	ANY	White - B...	2000-07-14	13	40	2
579	TESTN...	TESTNAME	N	N	N	ANY	White - B...	2000-07-15	13	41	2
580	TESTN...	TESTNAME	N	N	N	ANY	White - Iri...	2000-07-16	13	42	2
581	TESTN...	TESTNAME	N	N	N	ANY	Banglade...	2000-07-17	13	43	2
582	TESTN...	TESTNAME	N	N	N	ANY	White - B...	2000-07-18	13	44	2
583	TESTN...	TESTNAME	N	N	N	ANY	White - B...	2000-07-19	13	45	2
584	TESTN...	TESTNAME	N	N	N	ANY	White - B...	2000-07-20	13	46	2
585	TESTN...	TESTNAME	N	N	N	ANY	White - B...	2000-07-21	13	47	2

Properties

Aggregate Status

Connection f:

- Elapsed time: 00:00:00,423
- Finish time: 22/02/2016 13:30:48
- Name: STUDENTDEV
- Rows returned: 585
- Start time: 22/02/2016 13:30:48
- State: Open

Connection

Connection n: STUDENTDEV (dn.ingleton)

Connection Details

Connection e: 00:00:00,423

Connection f: 22/02/2016 13:30:48

Connection r: 585

Connection s: 22/02/2016 13:30:48

Connection t: Open

Display name: STUDENTDEV

Login name: dn.ingleton

Server name: STUDENTDEV

Server version: 10.50.1600

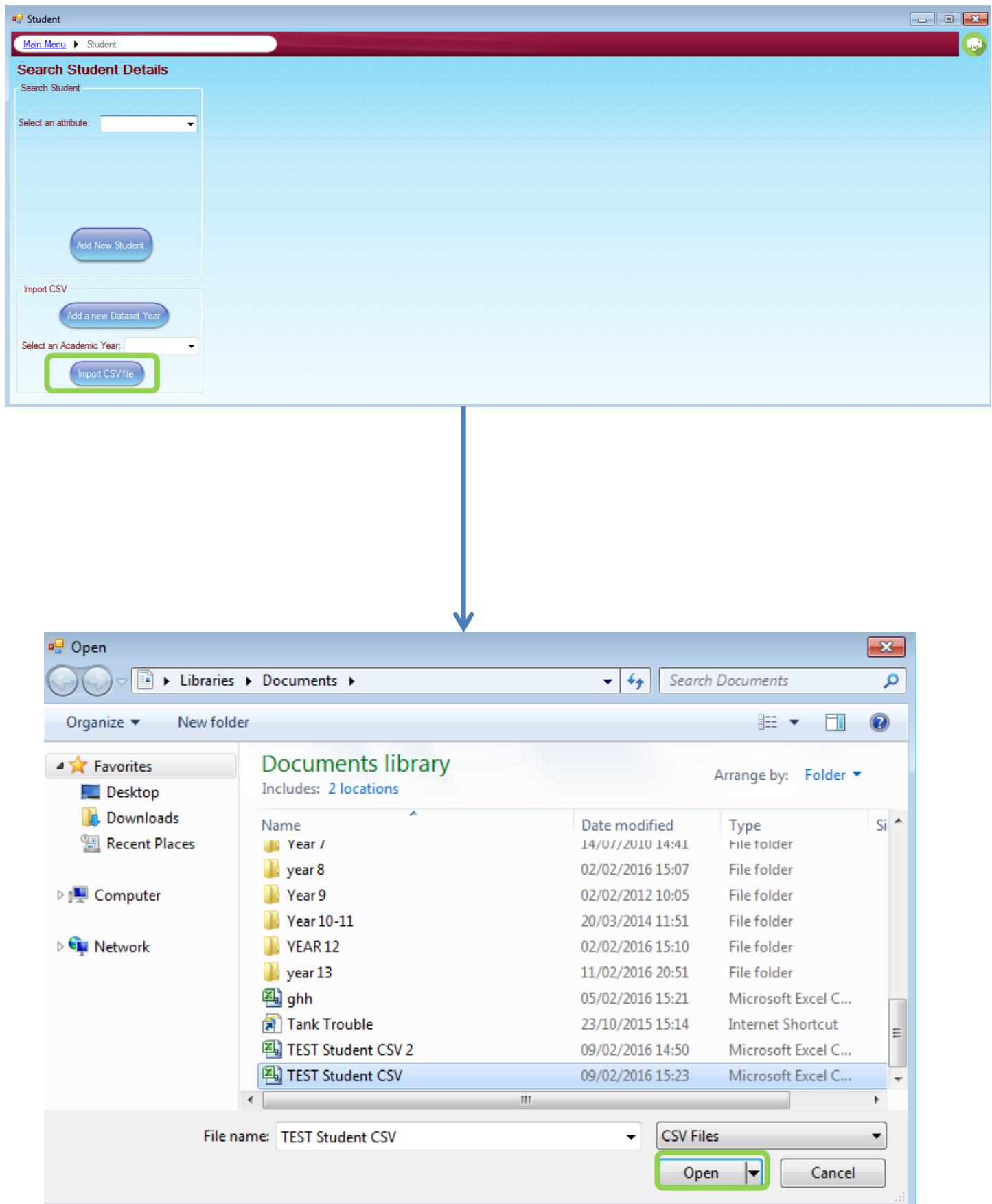
Session Tracit

SPID: 67

Name

The name of the connection.

Reference 10





Year 11 Dynamic Grade data 

ExecuteNonQuery requires an open and available Connection. The connection's current state is closed.

OK

Reference 11

Test illustrating the adding of a single student into the database.

Add Student

Main Menu > Student > Add Student

Add Student Details

Firstname:	david
Surname:	james
Gender:	F
Ethnicity:	White
Attendance(%):	98.7
Date Of Birth:	20 November 1997
Special Needs Status(SNS):	N
Entitled To Free School Meals(FSM):	Y
Form:	13W
Current School Year:	13
Exams Year:	2015

Add

Year 11 Dynamic Grade data

i Record Added to Database

OK

Evidence of student record being inserted into SQL server database.

The screenshot shows the Microsoft SQL Server Management Studio interface. In the center, there is a results grid titled 'Results' showing a list of student records. One specific row is highlighted with a green border, corresponding to the last record in the table:

StudentID	Firstname	Surname	Gender	FSM	SNS	Ethnicity	Attendance	DateOfBirth	Form	Year	DatasetID
547	Anna	WARD	F	N	K	White - British	100	1998-06-16	13W	13	1
548	Alice	WARD	F	N	N	White - British	91.7	1997-03-02	13T	13	1
549	Alice	WARDROP	F	N	N	White - British	100	1998-07-23	13A	13	1
550	Alexander	WATTS	M	N	N	White - British	96.7	1997-10-23	13C	13	1
551	Alec	WILSON	M	N	N	White - British	100	1997-11-19	13W	13	1
552	Alastair	WINSON-B...	M	N	N	White - British	96.7	1998-08-22	13W	13	1
553	Adam	WOOD	M	N	N	White - British	100	1998-07-10	13W	13	1
554	Achusla	WORTH	F	N	N	White - British	98.3	1998-02-19	13R	13	1
555											
556	david	james	F	Y	N	White	98.7	1997-11-20	13W	13	3

The status bar at the bottom of the screen displays the message "Query executed successfully." and other connection details.

Reference 12

This test illustrates what happens when a user attempts to insert a record but some fields are blank, it also shows what happens when a user attempts to insert a record again but completes previously incomplete fields. Note how the previously incomplete fields background colours change from red to white.

The screenshot displays two windows of a software application titled "Add Student".

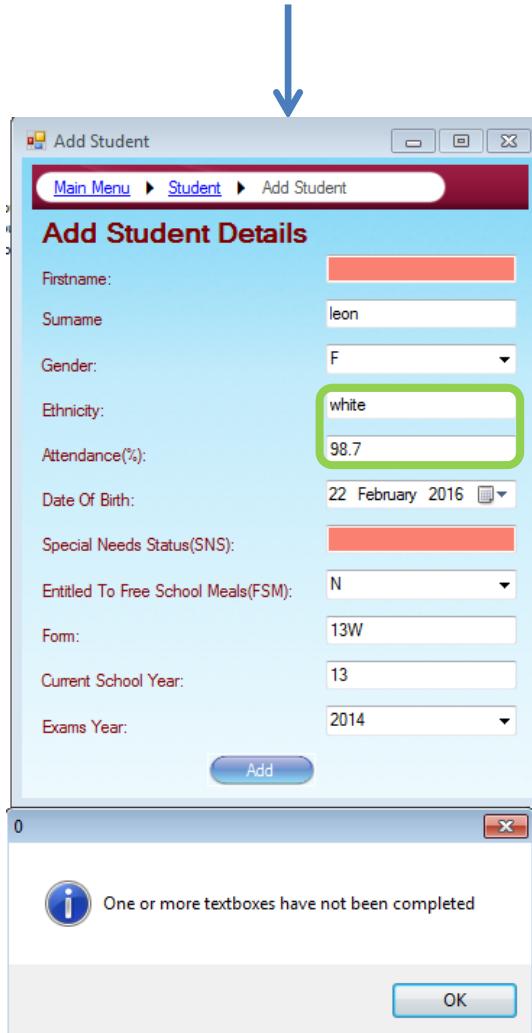
First Window (Top):

- Firstname: [Empty]
- Surname: leon
- Gender: F
- Ethnicity: [Empty]
- Attendance(%): [Empty]
- Date Of Birth: 22 February 2016
- Special Needs Status(SNS): [Empty]
- Entitled To Free School Meals(FSM): N
- Form: 13W
- Current School Year: 13
- Exams Year: 2014

Second Window (Bottom):

- Firstname: [Red]
- Surname: leon
- Gender: F
- Ethnicity: [Red]
- Attendance(%): [Red]
- Date Of Birth: 22 February 2016
- Special Needs Status(SNS): [Red]
- Entitled To Free School Meals(FSM): N
- Form: 13W
- Current School Year: 13
- Exams Year: 2014

A blue arrow points from the first window down to the second window. A green rounded rectangle highlights the "Add" button in both windows. A message box at the bottom of the second window contains the text: "One or more textboxes have not been completed".



Reference 13

This test illustrates the process involved with deleting a student's record.

The screenshot shows two windows related to student management:

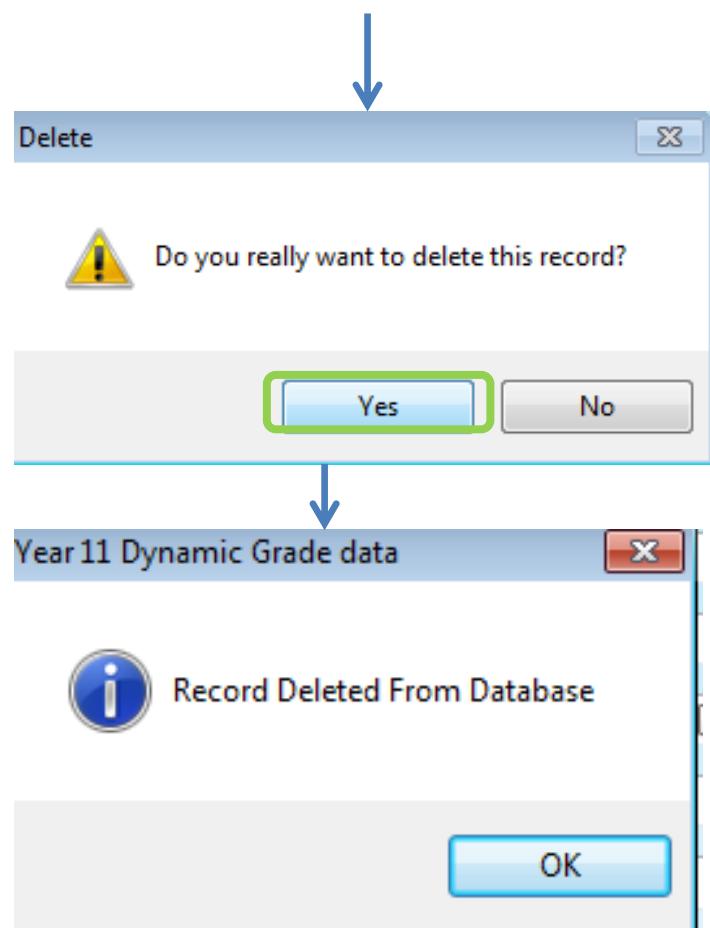
Search Student Details (Top Window):

- Search Student section:
 - Select an attribute: Name
 - Name: david
 - Search button
 - Add New Student button
- Import CSV section:
 - Add a new Dataset Year button
 - Select an Academic Year dropdown
 - Import CSV file button
- Data grid table:

Firstname	Surname	Gender	FSM	SNS	Ethnicity	Attendance	DateOfBirth	Form	Year	Student Details
David	MARSHALL	M	N	N	White - British	96.6	19/04/2000	11D	11	Edit Details
David	JONES	M	N	N	Refused	96.6	30/04/2000	11F	11	Edit Details
David	HALLAM	M	N	N	White - British	98.3	21/10/1999	11H	11	Edit Details
david	james	F	Y	N	White	98.7	20/11/1997	13W	13	Edit Details

Edit Student (Bottom Window):

- Main Menu > Student > Edit Student
- Edit Student Details form:
 - Firstname: david
 - Surname: james
 - Gender: F
 - Ethnicity: White
 - Attendance(%): 98.7
 - Date Of Birth: 20 November 1997
 - Special Needs Status(SNS): N
 - Entitled To Free School Meals(FSM): Y
 - Form: 13W
 - Current School Year: 13
- Buttons: Edit, Delete (highlighted)



The picture below shows that the record has actually been deleted from the SQL server database.

Microsoft SQL Server Management Studio

File Edit View Query Project Debug Tools Window Community Help

New Query Execute

Object Explorer

Connect

SQLQuery32.sql...ingleton (114) SQLQuery31.sql...leton (110)* SQLQuery30.sql...ingleton (117) script.sql - STU...ingleton (51)

***** Script for SelectTopNRows command from SSMS *****
SELECT TOP 1000 [StudentID]

StudentID	Firstname	Surname	Gender	FSM	SNS	Ethnicity	Attendance	DateOfBirth	Form	Year	DatasetID
533	Chloe	STANOVIEC	F	N	N	White - British	93.3	1998-01-02	13R	13	1
534	Charlotte	STARZA	F	N	N	White - British	93.3	1998-02-27	13T	13	1
535	Charlotte	STEWART	F	N	N	White - British	100	1998-03-27	13W	13	1
536	Charlie	STODDART	M	N	N	White - British	100	1997-10-09	13W	13	1
537	Catherine	SUTCLIFFE	F	N	N	White - British	96.7	1998-06-23	13A	13	1
538	Bridget	SYMONDS	F	N	N	White - British	98.3	1998-06-11	13B	13	1
539	Billy	TOWNSEND	M	N	N	White - British	95	1997-12-09	13W	13	1
540	Billy	TOZER	M	N	N	White - British	100	1998-02-28	13C	13	1
541	Bethany	TYMON	F	N	N	White - British	100	1998-04-18	13C	13	1
542	Beth	TYSON	F	N	N	White - British	96.7	1997-11-20	13D	13	1
543	Ben	VAN	M	N	N	Any other White background	91.7	1998-03-05	13W	13	1
544	Ben	VARIN	M	N	N	Any other White background	98.3	1998-03-05	13M	13	1
545	Becca	VIGH-BOL...	F	N	K	White - British	91.7	1997-12-23	13R	13	1
546	Annie	WALKER	F	N	N	White - British	81.7	1997-10-03	13T	13	1
547	Anna	WARD	F	N	K	White - British	100	1998-06-16	13W	13	1
548	Alice	WARD	F	N	N	White - British	91.7	1997-03-02	13T	13	1
549	Alice	WARDROP	F	N	N	White - British	100	1998-07-23	13A	13	1
550	Alexander	WATTS	M	N	N	White - British	96.7	1997-10-23	13C	13	1
551	Alec	WILSON	M	N	N	White - British	100	1997-11-19	13W	13	1
552	Alastair	WINSON-B...	M	N	N	White - British	96.7	1998-08-22	13W	13	1
553	Adam	WOOD	M	N	N	White - British	100	1998-07-10	13W	13	1
554	Achusa	WORTH	F	N	N	White - British	98.3	1998-02-19	13R	13	1
555	Abdullah	YATES	M	N	N	White - British	96.7	1997-10-23	13T	13	1

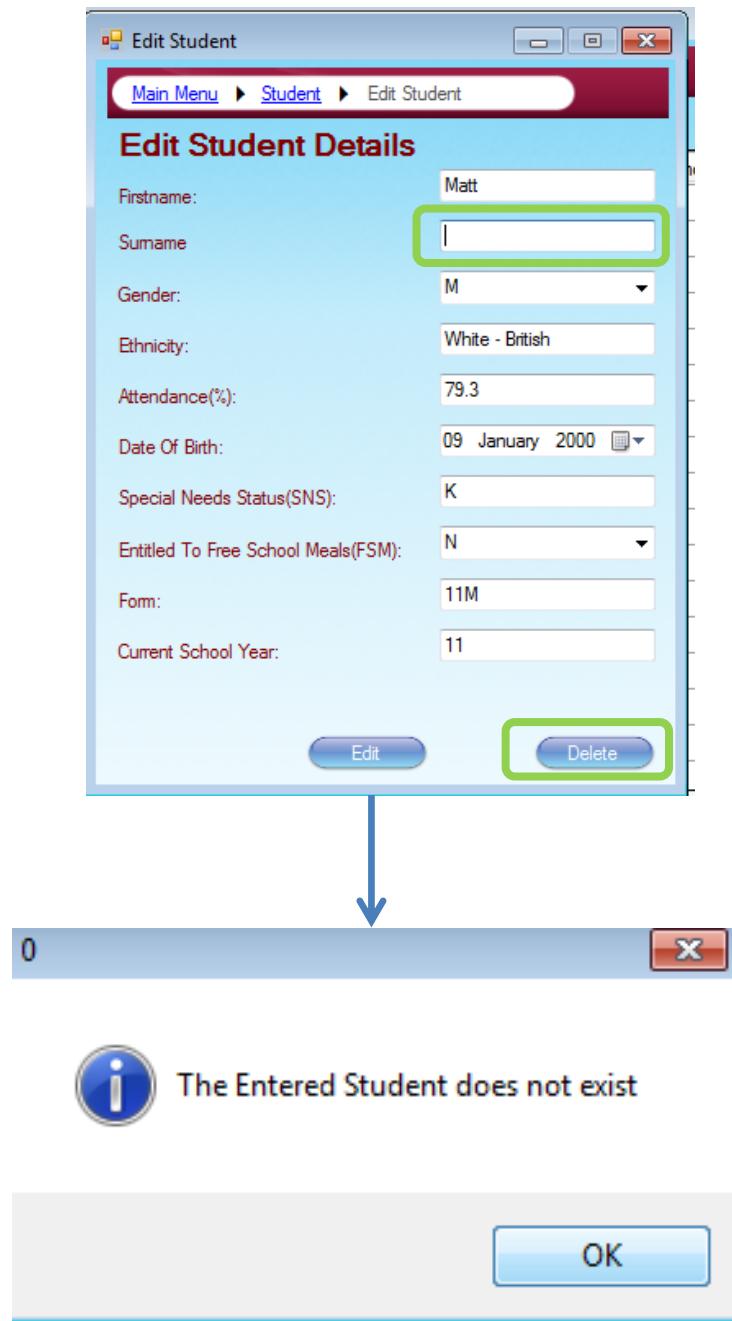
Query executed successfully.

Output

Ready

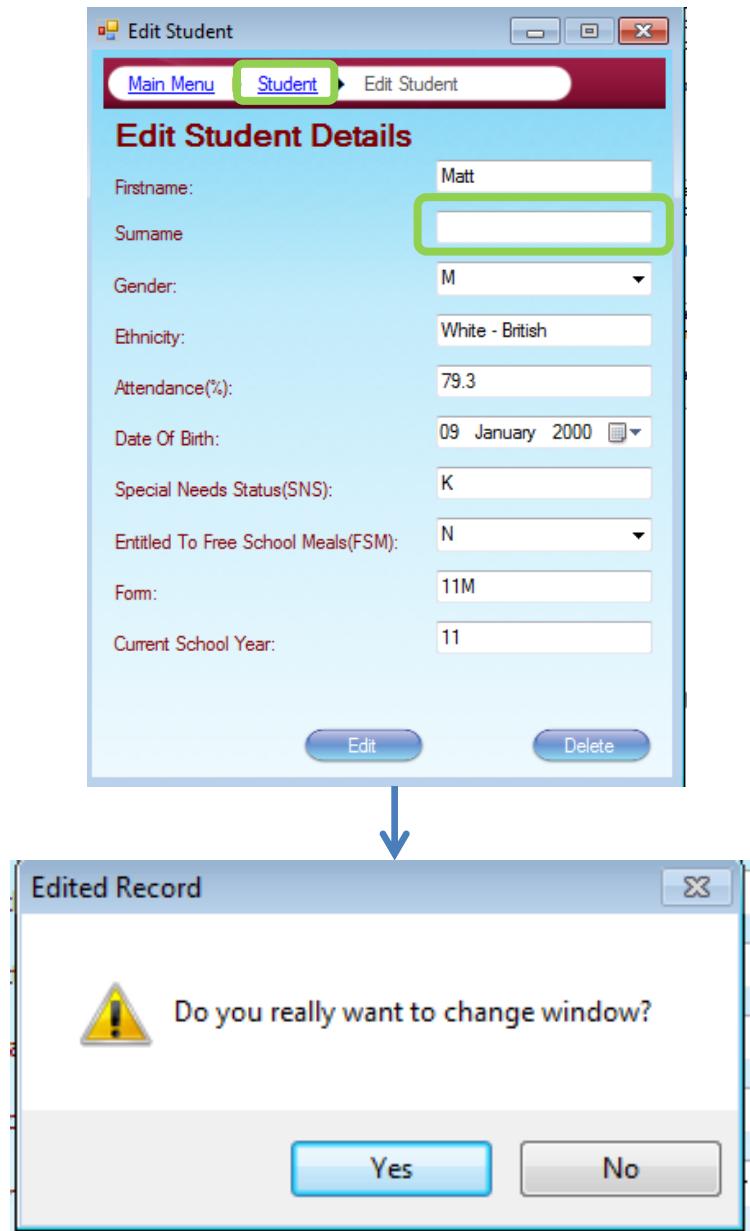
Reference 14

Test to show an attempt to delete a record which does not exist.



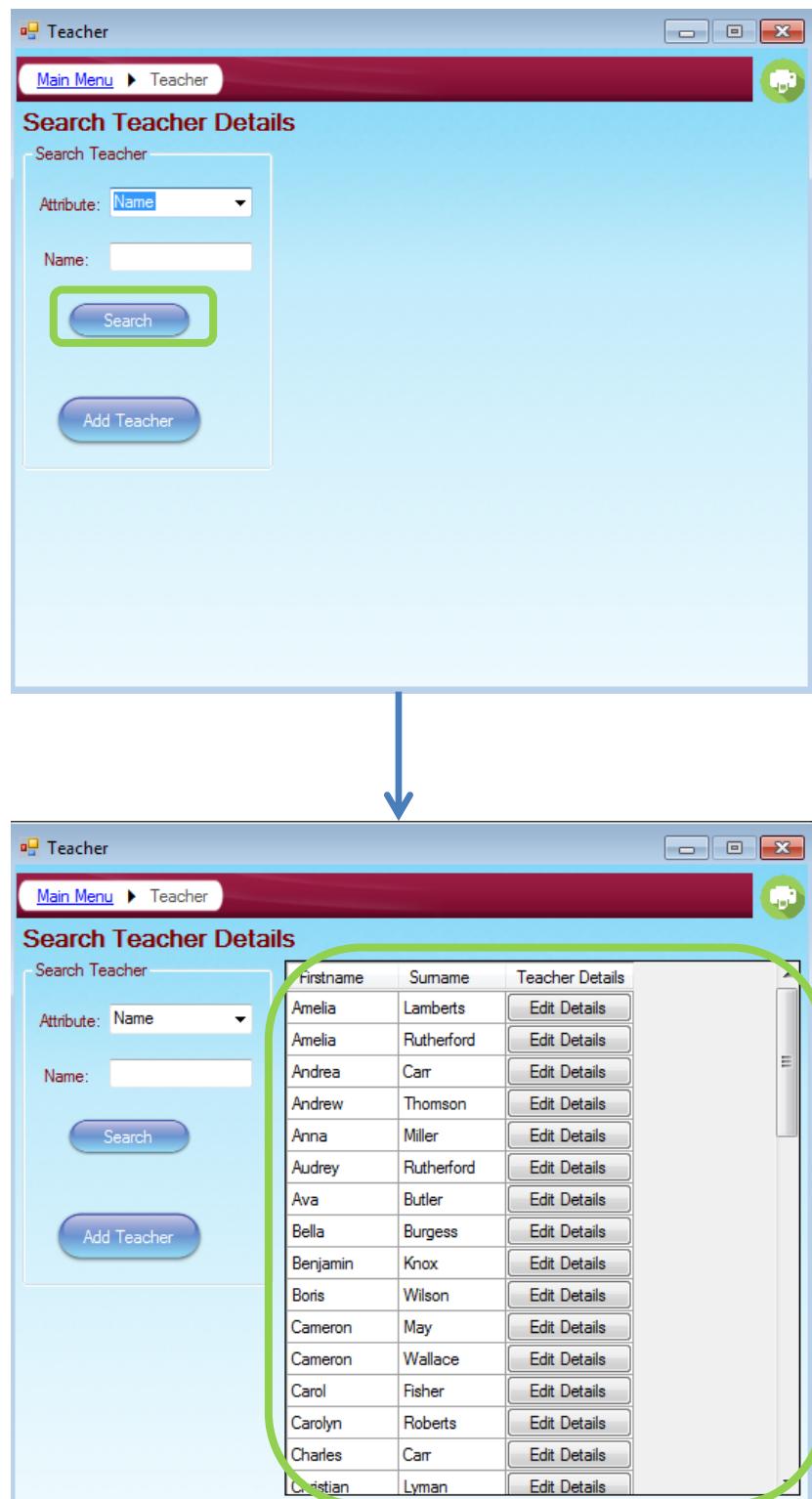
Reference 15

Test to show what happens when navigating away from the form after fields on the form have been edited.



Reference 16

Shows what happens when testing boundary data and a user has not chosen a filter option.



Reference 17

Teacher

Main Menu > Teacher

Search Teacher Details

Firstname	Surname	Teacher Details
Amelia	Lambers	Edit Details
Amelia	Rutherford	Edit Details
Andrea	Carr	Edit Details
Andrew	Thomson	Edit Details
Anna	Miller	Edit Details
Audrey	Rutherford	Edit Details
Ava	Butler	Edit Details
Bella	Burgess	Edit Details
Benjamin	Knox	Edit Details
Boris	Wilson	Edit Details
Cameron	May	Edit Details
Cameron	Wallace	Edit Details
Carol	Fisher	Edit Details
Carolyn	Roberts	Edit Details
Charles	Carr	Edit Details
Christian	Lyman	Edit Details

Attribute: Name

Name:

Search

Add Teacher

Add Teacher

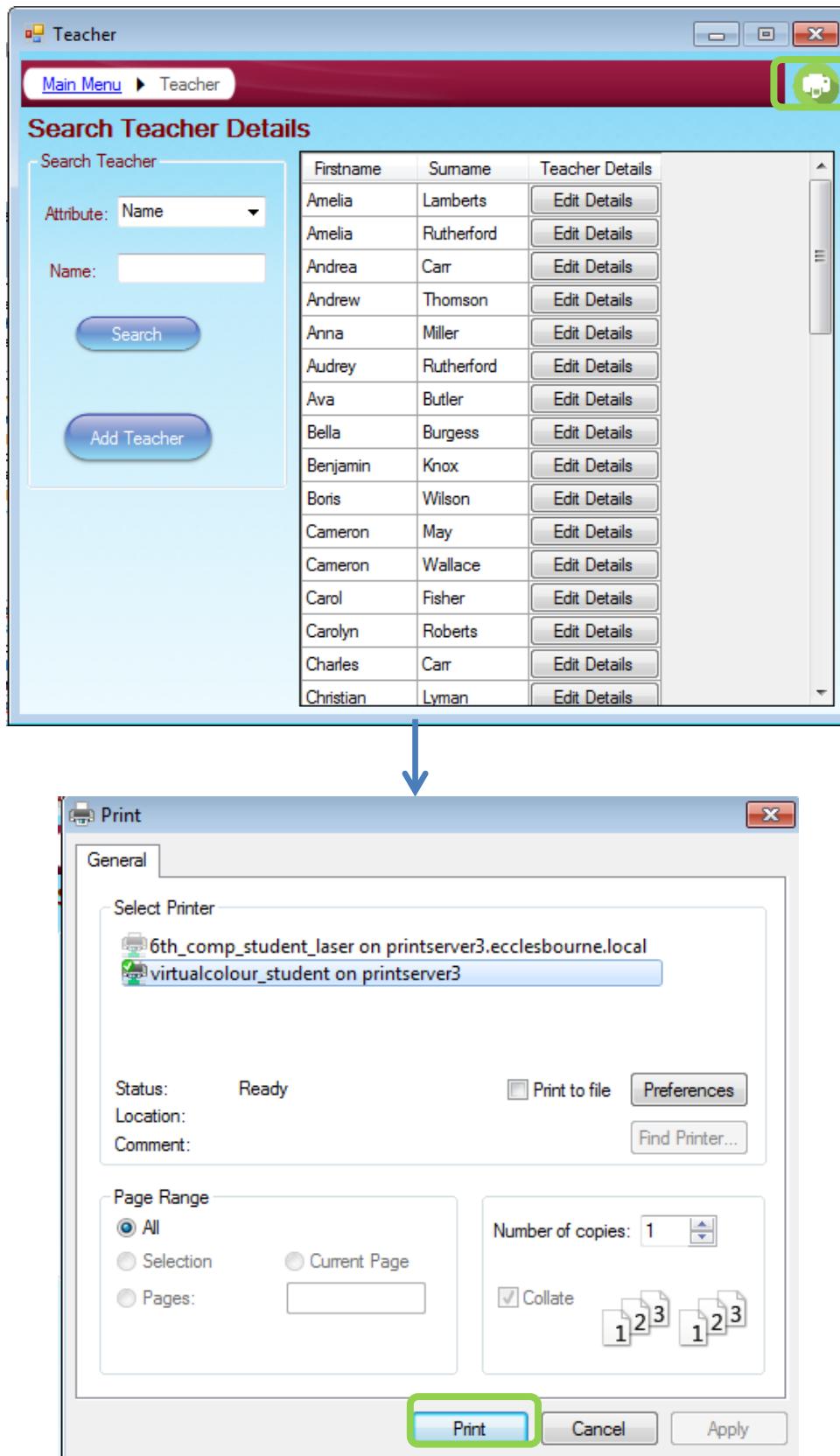
Main Menu > Teacher > Edit Teacher

Edit Teacher Details

Firstname:	<input type="text" value="Audrey"/>
Surname:	<input type="text" value="Rutherford"/>
<input type="button" value="Edit"/>	<input type="button" value="Delete"/>

Reference 18

Shows what happens when a user tries to print a forms selection. I will put a scanned copy of the printed output on the next page.



Main Menu ▶ Teacher

Search Teacher Details

Search Teacher

Attribute: Name

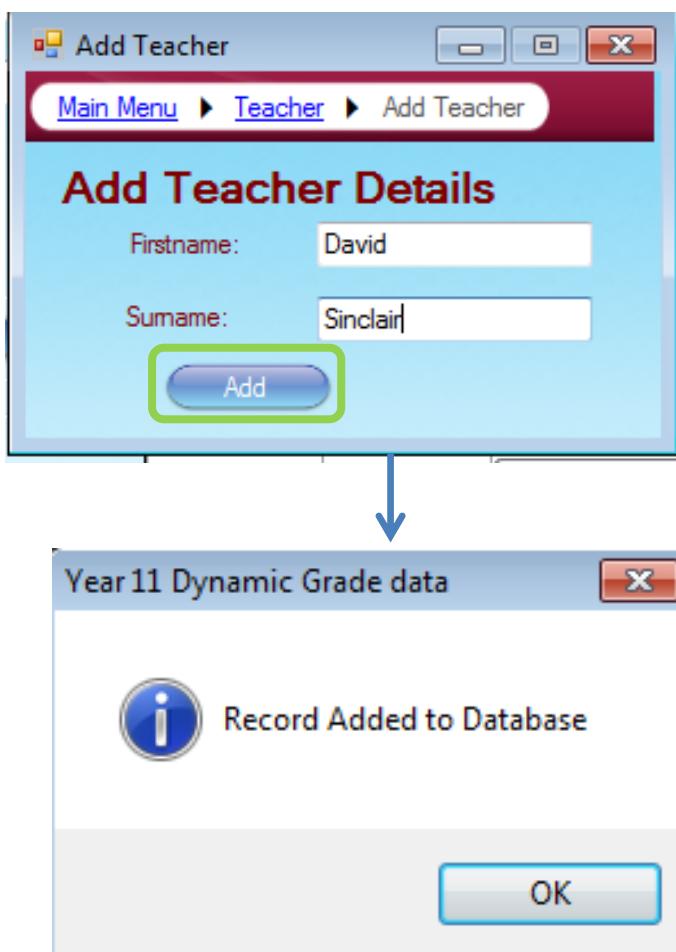
Name:

Search

Add Teacher

Firstname	Surname	Teacher Details
Amelia	Lamberts	Edit Details
Amelia	Rutherford	Edit Details
Andrea	Car	Edit Details
Andrew	Thomson	Edit Details
Amina	Miller	Edit Details
Audrey	Rutherford	Edit Details
Ava	Butler	Edit Details
Bella	Burgess	Edit Details
Benjamin	Knox	Edit Details
Boris	Wilson	Edit Details
Cameron	May	Edit Details
Cameron	Wallace	Edit Details
Carol	Foster	Edit Details
Carolyn	Roberts	Edit Details
Charles	Car	Edit Details
Christian	Luman	Edit Details

Reference 19



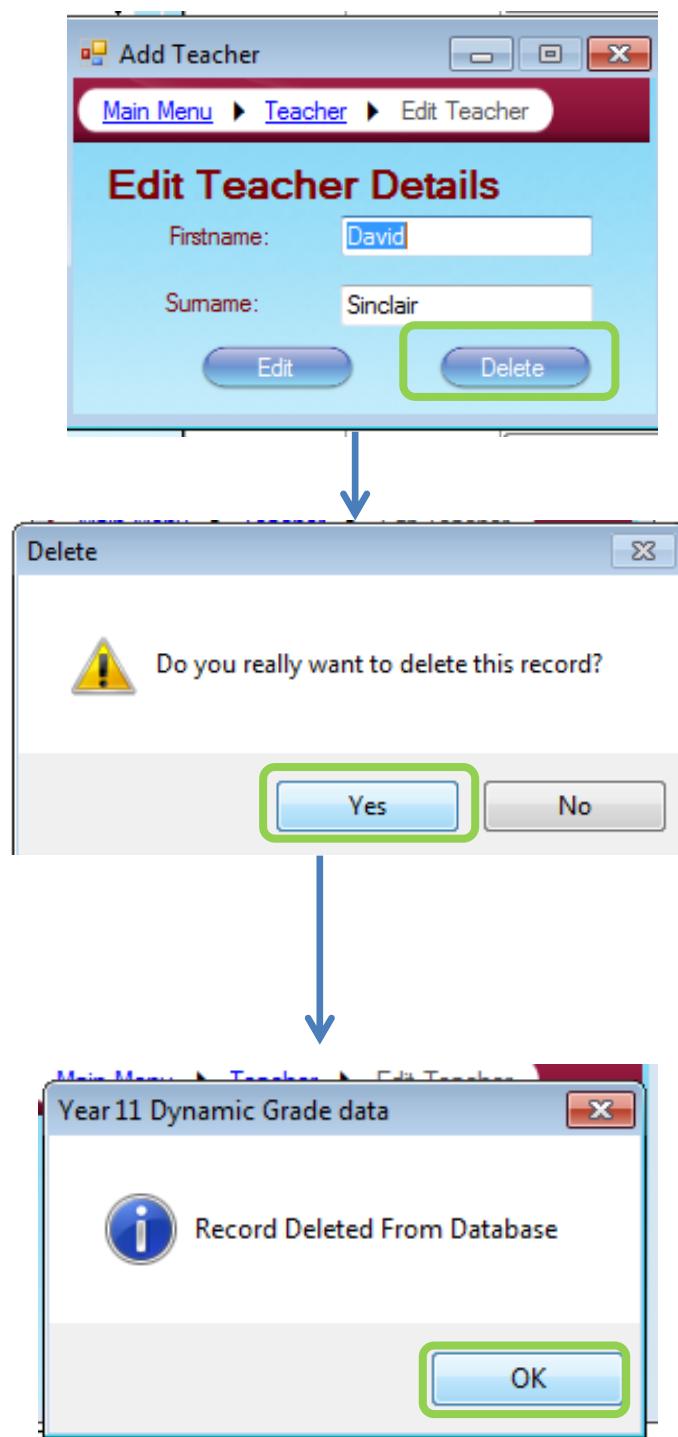
Evidence that record has been inserted successfully into the SQL server database.

	TeacherID	LoginID	Firstname	Surname
42	42	42	Jan	Bower
43	43	43	Carol	Fisher
44	44	44	Boris	Wilson
45	45	45	Gavin	Parr
46	46	46	Anna	Miller
47	47	47	Kevin	Davies
48	48	48	David	King
49	49	49	Theresa	Robertson
50	50	50	Joe	Campbell
51	51	NULL	David	Sinclair

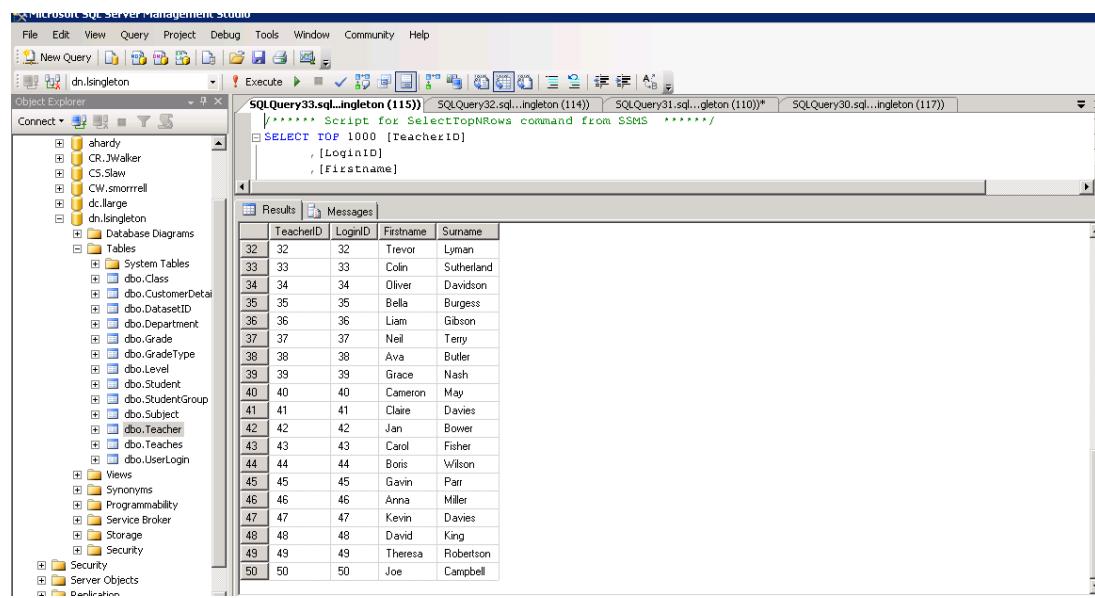
Query executed successfully. STUDENTDEV (10.50 RTM) dn.lsingleton (115) dn.lsingleton 00:00:00 51 rows

Reference 20

Test which shows the process of deleting a teacher from the system.



Evidence that record has been deleted successfully into the SQL server database.



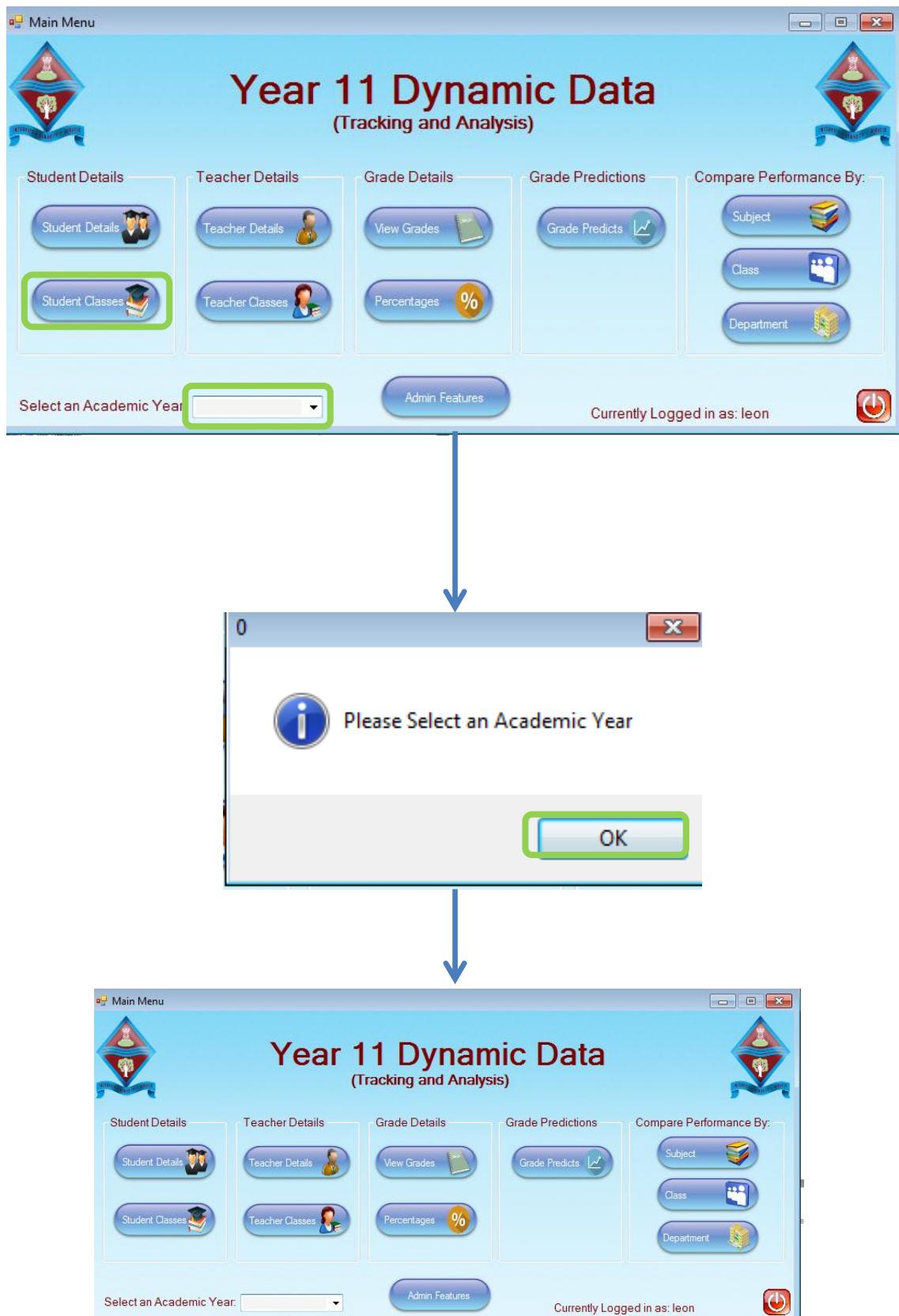
The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists several databases: ahardy, CR.JWalker, CS.Slaw, CW.smorrell, dc.llarge, dn.singleton, and others. The dn.singleton database is selected. The Results tab in the center displays the output of a SQL query:

```
SELECT TOP 1000 [TeacherID]
      , [LoginID]
      , [Firstname]
```

TeacherID	LoginID	Firstname	Surname
32	32	Trevor	Lyman
33	33	Colin	Sutherland
34	34	Oliver	Davidson
35	35	Bella	Burgess
36	36	Liam	Gibson
37	37	Neil	Terry
38	38	Ava	Buller
39	39	Grace	Nash
40	40	Cameron	May
41	41	Clair	Davies
42	42	Jan	Bower
43	43	Carol	Fisher
44	44	Boris	Wilson
45	45	Gavin	Parr
46	46	Anna	Miller
47	47	Kevin	Davies
48	48	David	King
49	49	Theresa	Robertson
50	50	Joe	Campbell

Reference 21

Test which shows what happens when clicking a button that requires an academic year to be selected.



Reference 22

Test to show the functionality of the data grid when the view classes button is clicked.

Student Classes

View a Students Classes

Select an attribute: Name

Name: will

Search

Or

Add and Remove Students from classes

First you must Select the class:

Select an attribute:

Search

Firstname	Surname	Gender	DateOfBirth	Form	Year	View a Students Classes
William	ABERCROMBIE	M	31/01/1997	13R	13	<button>View Classes</button>
William	AKRAM	M	11/02/1998	13M	13	<button>View Classes</button>
William	ALLISON	M	24/05/1998	13B	13	<button>View Classes</button>
Will	ALTOFT	M	13/09/1997	13C	13	<button>View Classes</button>

Student Classes

View a Students Classes

Select an attribute: Name

Name: will

Search

Or

Add and Remove Students from classes

First you must Select the class:

Select an attribute:

Search

ClassID	Firstname	Surname	ClassGroup	Subject	Department	Year
152	Will	ALTOFT	Class Group 152	Business Studies	Technology	2015
155	Will	ALTOFT	Class Group 155	English Language	English	2015
160	Will	ALTOFT	Class Group 160	English Literature	English	2015
165	Will	ALTOFT	Class Group 165	Food Technology	Technology	2015
166	Will	ALTOFT	Class Group 166	Geography	Humanities	2015
171	Will	ALTOFT	Class Group 171	Health And Socia...	Humanities	2015
172	Will	ALTOFT	Class Group 172	History	Humanities	2015
177	Will	ALTOFT	Class Group 177	ICT	Technology	2015
183	Will	ALTOFT	Class Group 183	Maths	Maths	2015
193	Will	ALTOFT	Class Group 193	Religious Studies	Humanities	2015

Reference 23

Test to show the functionality of the data grid when the view students button is clicked.

Student Classes

View a Students Classes

Select an attribute:

Search

Or

Add and Remove Students from classes

First you must Select the class:

Select an attribute: Department

Department: Physical

Search

ClassGroup	Subject	Department	Year	Firstname	Surname	View Current Students	Add a Student to this Class
Class Group 65	Physical Education	Physical	2013	Stewart	Murray	View Students	Add Student
Class Group 64	Physical Education	Physical	2013	Trevor	Burgess	View Students	Add Student

Student Classes

View a Students Classes

Select an attribute:

Search

Or

Add and Remove Students from classes

First you must Select the class:

Select an attribute: Department

Department: Physical

Search

Firstname	Surname	Form	ClassGroup	Subject	Department	Year	Remove this Student from this class
Alex	STEWART	11S	Class Group 64	Physical Education	Physical	2013	Remove Student
Annie	POPE	11L	Class Group 64	Physical Education	Physical	2013	Remove Student
Ashley	PARR	11J	Class Group 64	Physical Education	Physical	2013	Remove Student
Ben	MCCORMACK	11H	Class Group 64	Physical Education	Physical	2013	Remove Student
Ben	SPENCER	11L	Class Group 64	Physical Education	Physical	2013	Remove Student
Bethany	MEYNELL	11B	Class Group 64	Physical Education	Physical	2013	Remove Student
Brooke	OWENS	11F	Class Group 64	Physical Education	Physical	2013	Remove Student
Charlotte	ROOME	11S	Class Group 64	Physical Education	Physical	2013	Remove Student
Christopher	MERRIMAN	11J	Class Group 64	Physical Education	Physical	2013	Remove Student
Eleanor	HEWING	11B	Class Group 64	Physical Education	Physical	2013	Remove Student
Francesca	REEVES	11L	Class Group 64	Physical Education	Physical	2013	Remove Student
Freya	PARKER	11M	Class Group 64	Physical Education	Physical	2013	Remove Student
George	LIVINGSTON	11F	Class Group 64	Physical Education	Physical	2013	Remove Student
George	WATTS	11D	Class Group 64	Physical Education	Physical	2013	Remove Student
Hannah	JARIC	11B	Class Group 64	Physical Education	Physical	2013	Remove Student
Jacob	PRICE	11M	Class Group 64	Physical Education	Physical	2013	Remove Student

Reference 24

Test to show the functionality of the data grid when the add student button is clicked.

Student Classes

Main Menu > Student Classes

Student Classes

View a Students Classes

Select an attribute:

Search

Or

Add and Remove Students from classes

First you must Select the class:

Select an attribute: Department

Department: Physical

Search

ClassGroup	Subject	Department	Year	Firstname	Surname	View Current Students	Add a Student to this Class
Class Group 65	Physical Education	Physical	2013	Stewart	Murray	View Students	Add Student
Class Group 64	Physical Education	Physical	2013	Trevor	Burgess	View Students	Add Student

↓

Add Student Class

Main Menu > Student Classes > Add Student

Add Student to Class

Current Selected Class:

ClassGroup	Subject	Year
Class Group 64	Physical Education	2013

Search for a Student

Select an attribute:

Reference 25

Test to show the functionality of the data grid when the remove student button is clicked.

Student Classes

Main Menu > Student Classes

Student Classes

View a Students Classes

Select an attribute:

Search

Or

Add and Remove Students from classes

First you must Select the class:

Select an attribute: Department

Department: Physchical

Search

Firstname	Surname	Form	ClassGroup	Subject	Department	Year	Remove this Student from this class
Alex	STEWART	11S	Class Group 64	Physchical Education	Physchical	2013	<button>Remove Student</button>
Annie	POPE	11L	Class Group 64	Physchical Education	Physchical	2013	<button>Remove Student</button>
Ashley	PARR	11J	Class Group 64	Physchical Education	Physchical	2013	<button>Remove Student</button>
Ben	MCCORMACK	11H	Class Group 64	Physchical Education	Physchical	2013	<button>Remove Student</button>
Ben	SPENCER	11L	Class Group 64	Physchical Education	Physchical	2013	<button>Remove Student</button>
Bethany	MEYNELL	11B	Class Group 64	Physchical Education	Physchical	2013	<button>Remove Student</button>
Brooke	OWENS	11F	Class Group 64	Physchical Education	Physchical	2013	<button>Remove Student</button>
Charlotte	ROOME	11S	Class Group 64	Physchical Education	Physchical	2013	<button>Remove Student</button>
Christopher	MERRIMAN	11J	Class Group 64	Physchical Education	Physchical	2013	<button>Remove Student</button>
Eleanor	HEWING	11B	Class Group 64	Physchical Education	Physchical	2013	<button>Remove Student</button>
Francesca	REEVES	11L	Class Group 64	Physchical Education	Physchical	2013	<button>Remove Student</button>
Freya	PARKER	11M	Class Group 64	Physchical Education	Physchical	2013	<button>Remove Student</button>
George	LIVINGSTON	11F	Class Group 64	Physchical Education	Physchical	2013	<button>Remove Student</button>
George	WATTS	11D	Class Group 64	Physchical Education	Physchical	2013	<button>Remove Student</button>
Hannah	JARIC	11B	Class Group 64	Physchical Education	Physchical	2013	<button>Remove Student</button>
Jacob	PRICE	11M	Class Group 64	Physchical Education	Physchical	2013	<button>Remove Student</button>

Remove Student Class

Main Menu > Student Classes > Remove Student

Remove Student From Class

Current Selected Class:

ClassGroup	Subject	Year
Class Group 64	Physchical Education	2013

Student Firstname: Alex

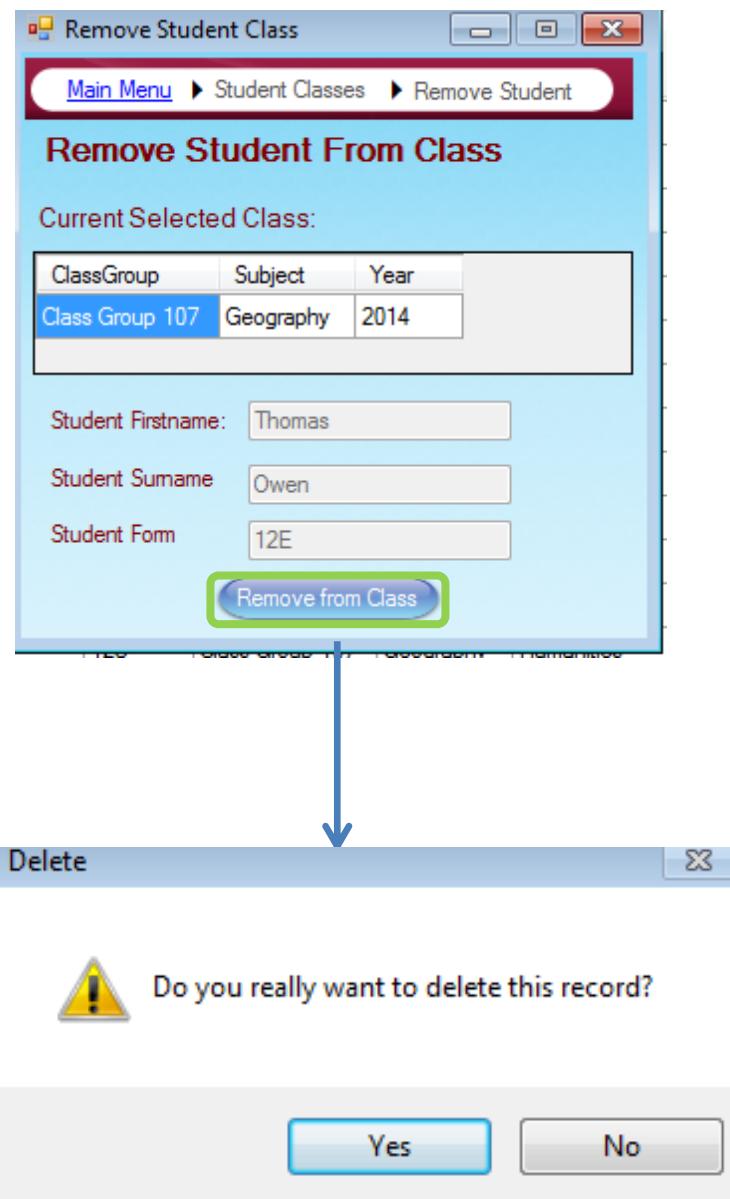
Student Surname: STEWART

Student Form: 11S

Remove from Class

Reference 26

This test shows what happens if a user tries to delete a student from a selected class.



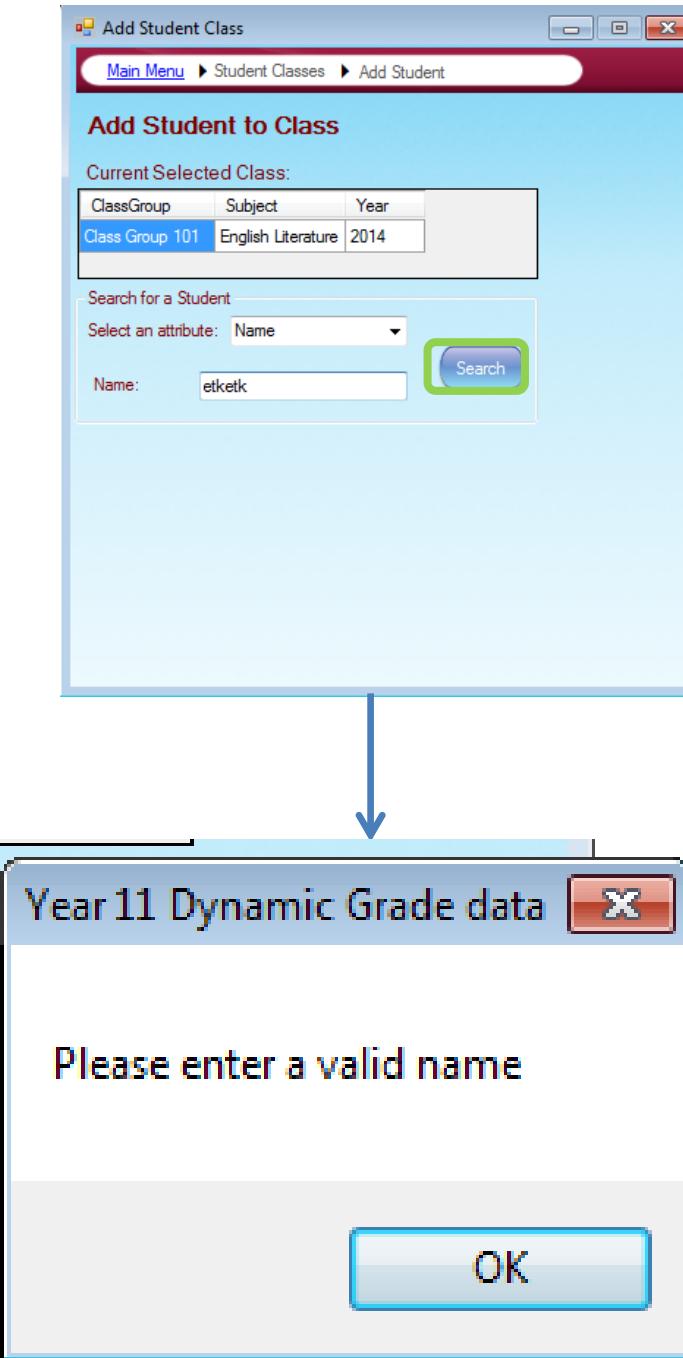
Reference 26 (continued)

The screen capture below shows that the Thomas Owen has been removed from the class correctly within the SQL server database. The Student ID of Thomas Owen was 214 but his record for this class has been removed, so he is no longer in the class.

StudentGroupId	ClassID	StudentID
1	107	335
2	107	339
3	107	341
4	107	342
5	107	343
6	107	347
7	107	349
8	107	350
9	107	351
10	107	352
11	107	353
12	107	354
13	107	361
14	107	364
15	107	365
16	107	367
17	107	368
18	107	372
19	107	373
20	107	374
21	107	375
22	107	379
23	107	380

Reference 27

This test evidence shows what happens when a user searches for an invalid name.



Reference 28

Add Student Class

Main Menu > Student Classes > Add Student

Add Student to Class

Current Selected Class:

ClassGroup	Subject	Year
Class Group 98	English Language	2014

Search for a Student

Select an attribute: Name

Name: will

Search

Results:

Firstname	Surname	Form	Year	Select
William	Muriel	12S	12	Select
Will	William	12A	12	Select
Rosie	Will	12S	12	Select

↓

Add Student Class

Main Menu > Student Classes > Add Student

Add Student to Class

Current Selected Class:

ClassGroup	Subject	Year
Class Group 98	English Language	2014

Search for a Student

Select an attribute: Name

Name: will

Search

Add Selected Student to Class

Firstname: Rosie

Surname: Will

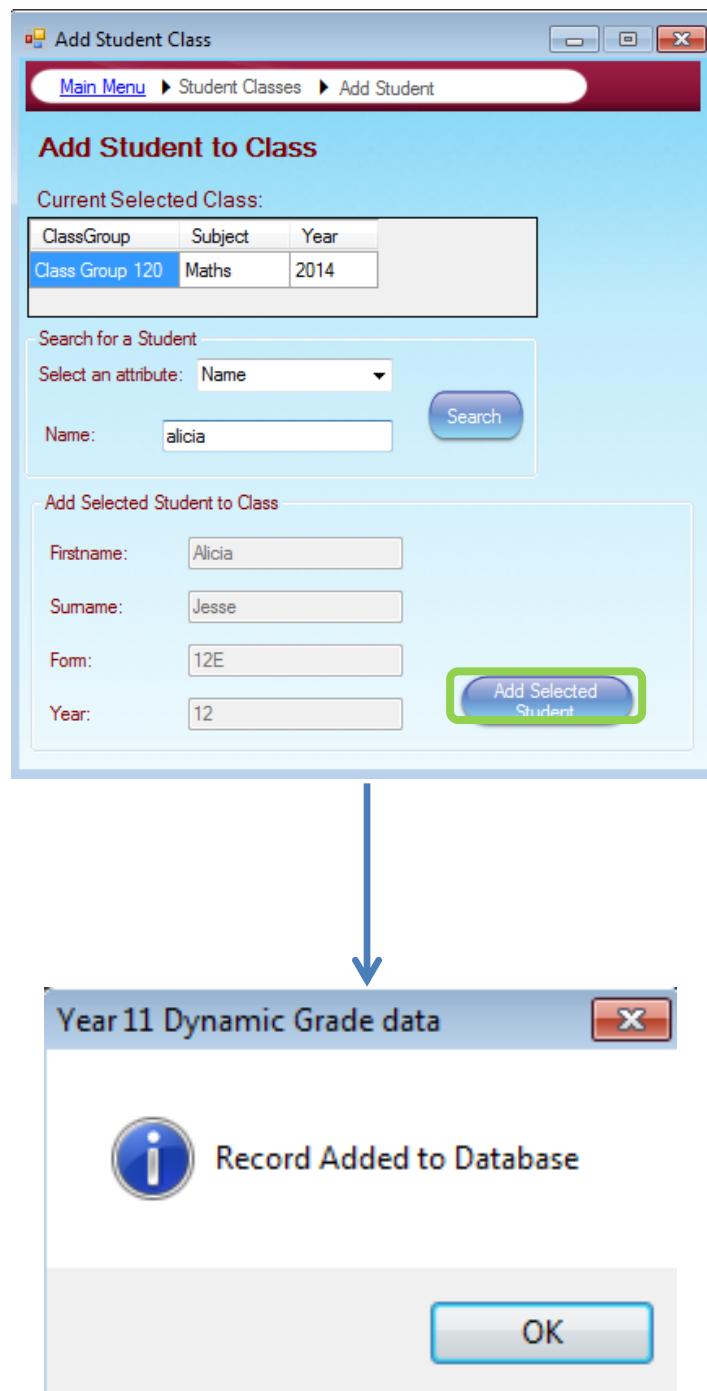
Form: 12S

Year: 12

Add Selected Student

Reference 29

Test evidence that shows adding a student to a class within the system.



Reference 29 (continued)

The screen capture below shows the actual student record being added to the class. Jesse's student ID is 341.

A screenshot of Microsoft SQL Server Management Studio (SSMS) showing a query results grid. The query is:

```
SELECT * FROM ClassStudent WHERE ClassID = 120
```

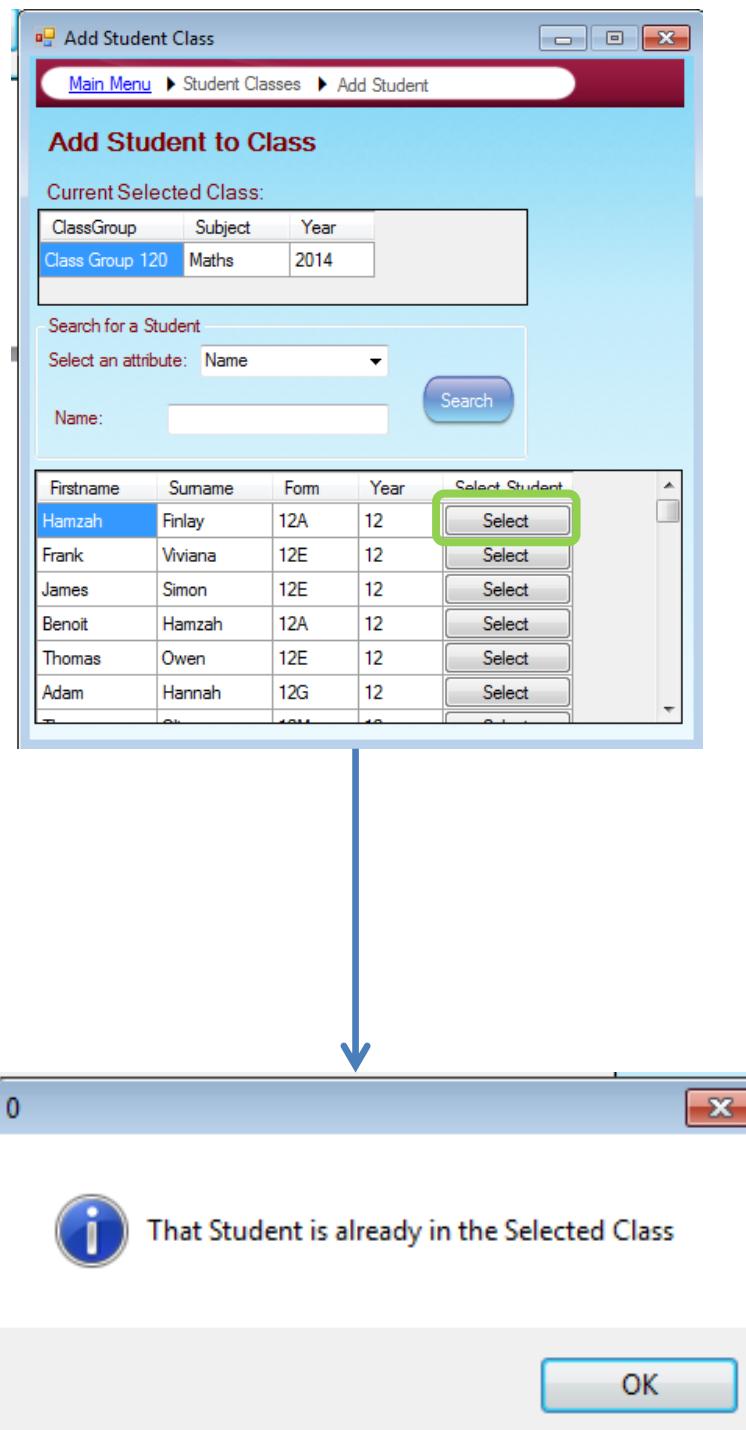
The results grid displays 30 rows of data:

	StudentGroupID	ClassID	StudentID
10	3359	120	219
11	3360	120	220
12	3361	120	221
13	3362	120	222
14	3363	120	223
15	3364	120	224
16	3365	120	226
17	3366	120	227
18	3367	120	228
19	3368	120	229
20	3369	120	230
21	3370	120	231
22	3371	120	232
23	3372	120	233
24	3373	120	234
25	3374	120	235
26	3375	120	236
27	3376	120	237
28	3377	120	238
29	3378	120	239
30	5652	120	341

The last row (StudentID 341) is highlighted with a green border. The status bar at the bottom of the SSMS window indicates "Query executed successfully."

Reference 30

For this test I have purposely selected a student that is already in the selected class to test my system validation.



Reference 31

Evidence to show the functionality of the data grid.

The image shows two windows of a software application. The top window is titled "Teacher Classes" and displays a data grid with one row of data. The bottom window is titled "Edit Classes" and contains a form for editing class details.

Teacher Classes Window:

- Search for a Class:
 - Select an attribute: Subject
 - Subject: Drama
- Search button
- Create a New Class:
 - Teacher Firstname: [empty]
 - Teacher Surname: [empty]
 - Class Group: [empty]
 - Subject: [empty]
 - Year: [empty]
- Create Class button
- Data Grid (1 row):

ClassGroup	Subject	Department	Year	Firstname	Surname	Action
Class Group 93	Drama	Arts	2014	Trevor	Burgess	Edit Class

Edit Classes Window:

- Main Menu > Teacher Classes > Edit Classes
- Edit Classes title
- Form fields:
 - Teacher Firstname: Trevor
 - Teacher Surname: Burgess
 - Class Group: Class Group 93
 - Subject: Drama
 - Year: 2014
- Edit Class and Delete buttons

A large blue arrow points from the "Edit Class" button in the Teacher Classes grid to the "Edit Class" button in the Edit Classes dialog.

Reference 32

Evidence showing the addition of a new valid class to the system.

Teacher Classes

Main Menu > Teacher Classes

Teacher Classes

Search for a Class

Select an attribute: Subject

Subject: Drama

Search

Create a New Class

Teacher Firstname: Trevor

Teacher Surname: Burgess

Class Group: TEX1

Subject: Textiles

Year: 2014

Create Class

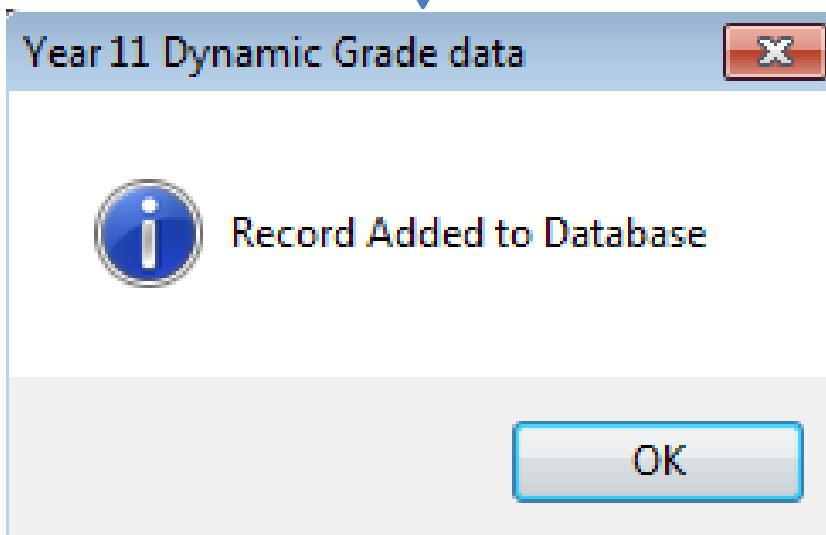
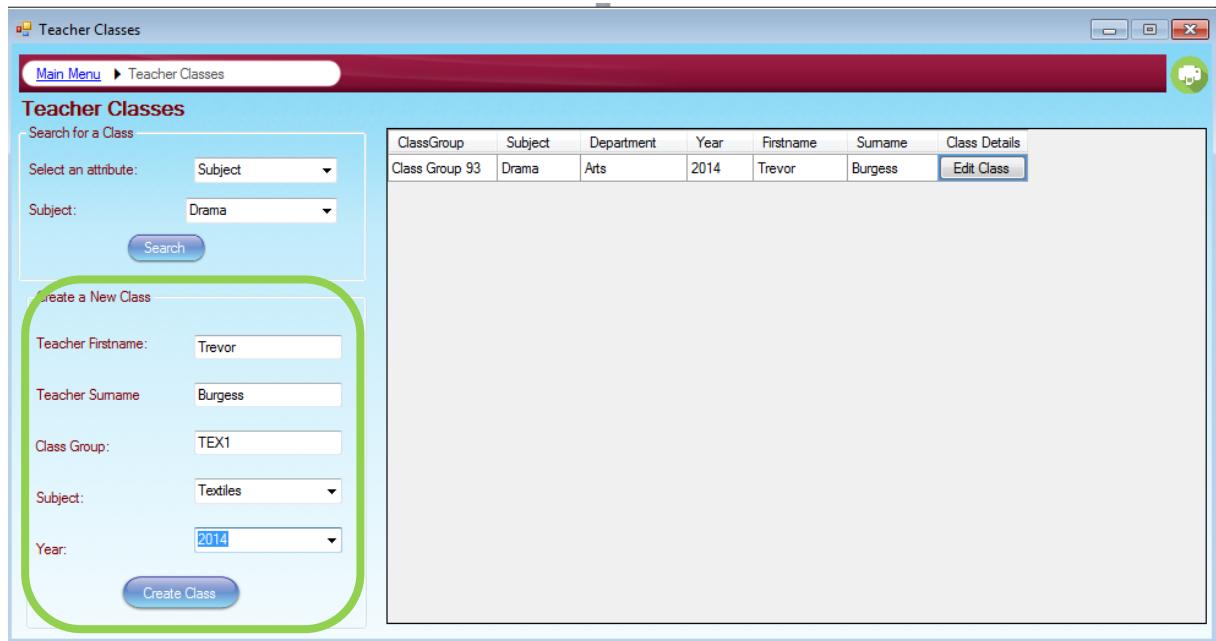
ClassGroup	Subject	Department	Year	Firstname	Surname	Class Details
Class Group 93	Drama	Arts	2014	Trevor	Burgess	Edit Class

↓

Year 11 Dynamic Grade data

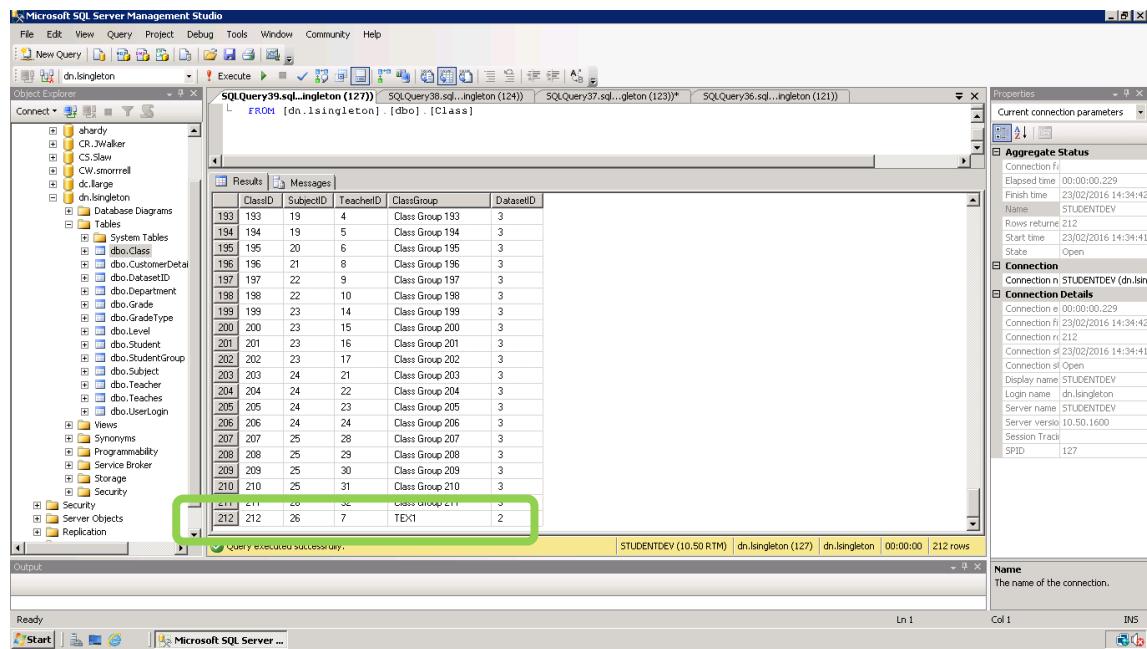
Record Added to Database

OK



Reference 32 (continued)

The screen capture below shows that the new class the user has entered has been successfully added to the SQL server database.



The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the database 'dn.singleton' is selected, showing various schemas like 'dbo', 'sys', and 'INFORMATION_SCHEMA'. In the 'Tables' node, there are several tables including 'Student', 'Subject', 'Teacher', 'Class', 'ClassGroup', and 'Dataset'. A query window titled 'SQLQuery39.sql...ingleton (127)' is open, displaying the following T-SQL code:

```
FROM [dn.singleton].[dbo].[Class]
```

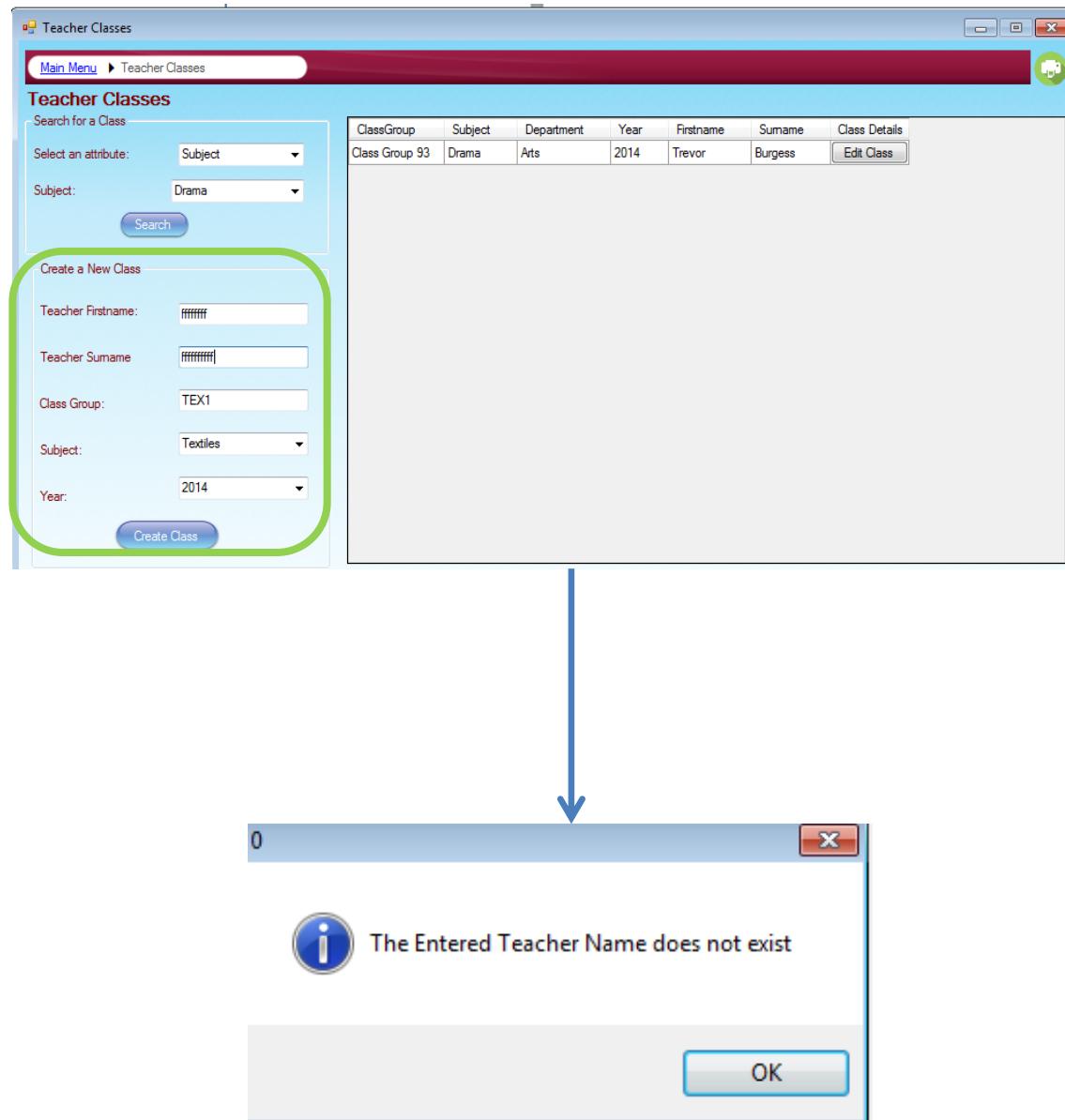
The results grid shows a list of rows from the 'Class' table. A green box highlights the last two rows, which were inserted by the user:

ClassID	SubjectID	TeacherID	ClassGroup	DatasetID	
193	193	19	4	Class Group 193	3
194	194	19	5	Class Group 194	3
195	195	20	6	Class Group 195	3
196	196	21	8	Class Group 196	3
197	197	22	9	Class Group 197	3
198	198	22	10	Class Group 198	3
199	199	23	14	Class Group 199	3
200	200	23	15	Class Group 200	3
201	201	23	16	Class Group 201	3
202	202	23	17	Class Group 202	3
203	203	24	21	Class Group 203	3
204	204	24	22	Class Group 204	3
205	205	24	23	Class Group 205	3
206	206	24	24	Class Group 206	3
207	207	25	28	Class Group 207	3
208	208	25	29	Class Group 208	3
209	209	25	30	Class Group 209	3
210	210	25	31	Class Group 210	3
211	211	26	32	Class Group 211	3
212	212	26	7	TEX1	2

The status bar at the bottom of the interface indicates: 'Query executed successfully.' and 'STUDENTDEV (10.50 RTM) dn.singleton (127) dn.singleton 00:00:00 212 rows'.

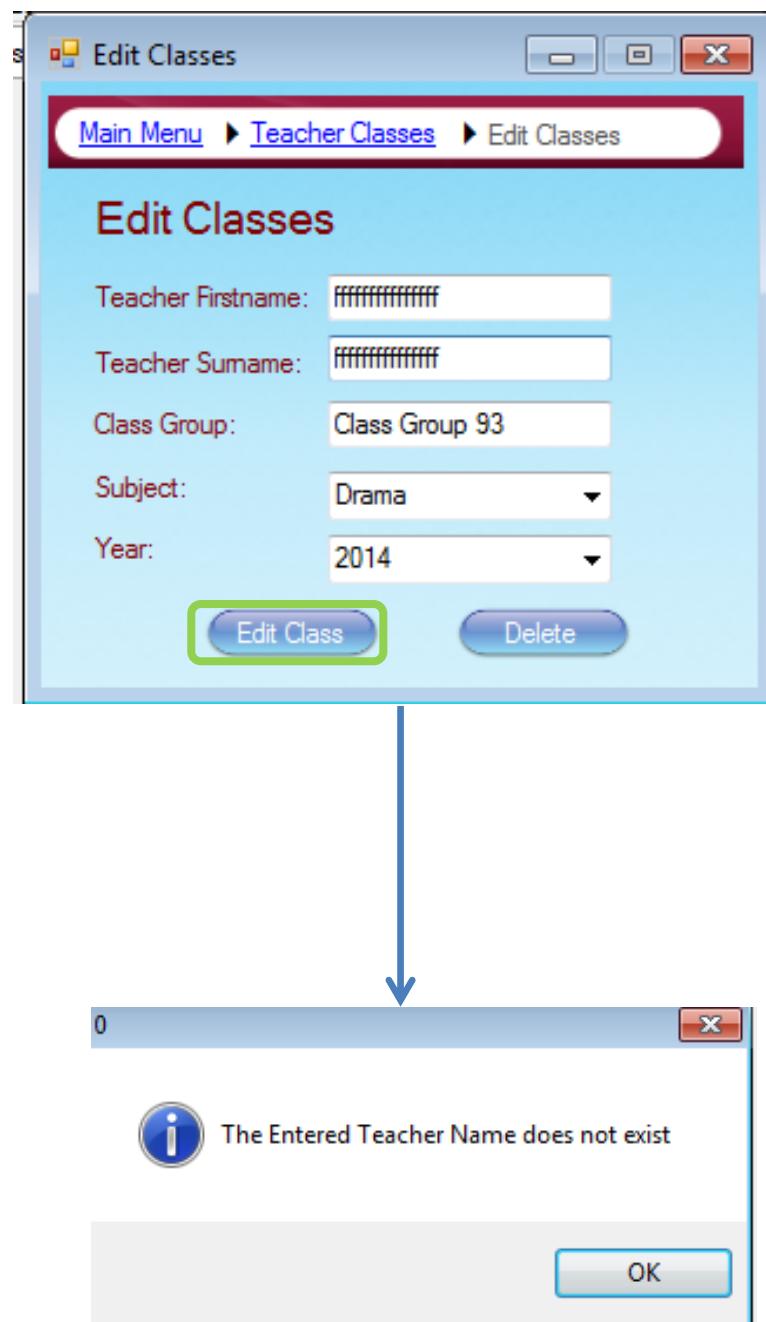
Reference 33

Evidence showing the attempted addition of a non-valid class to the system.



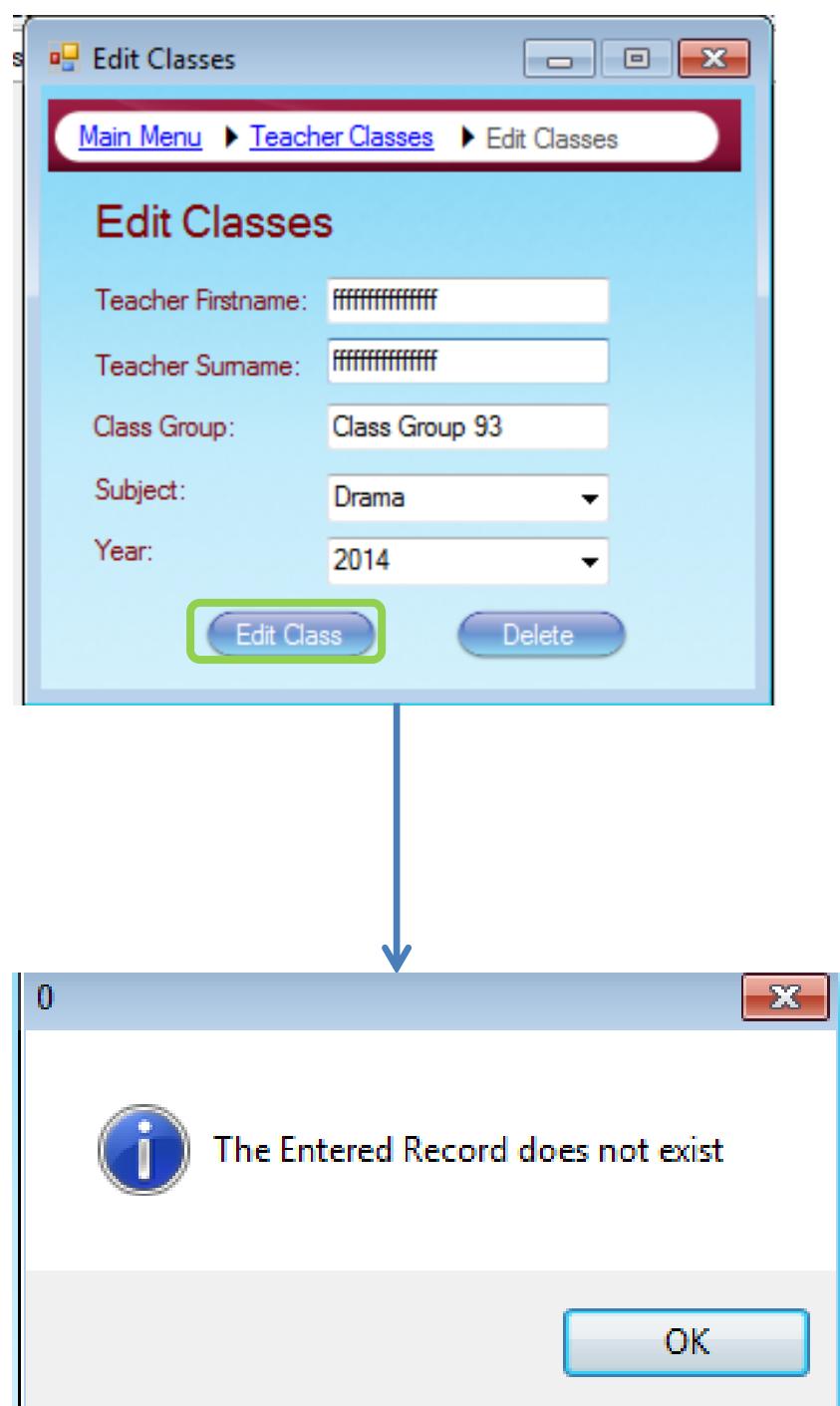
Reference 34

Evidence showing the attempted edit of a class with an invalid teacher.



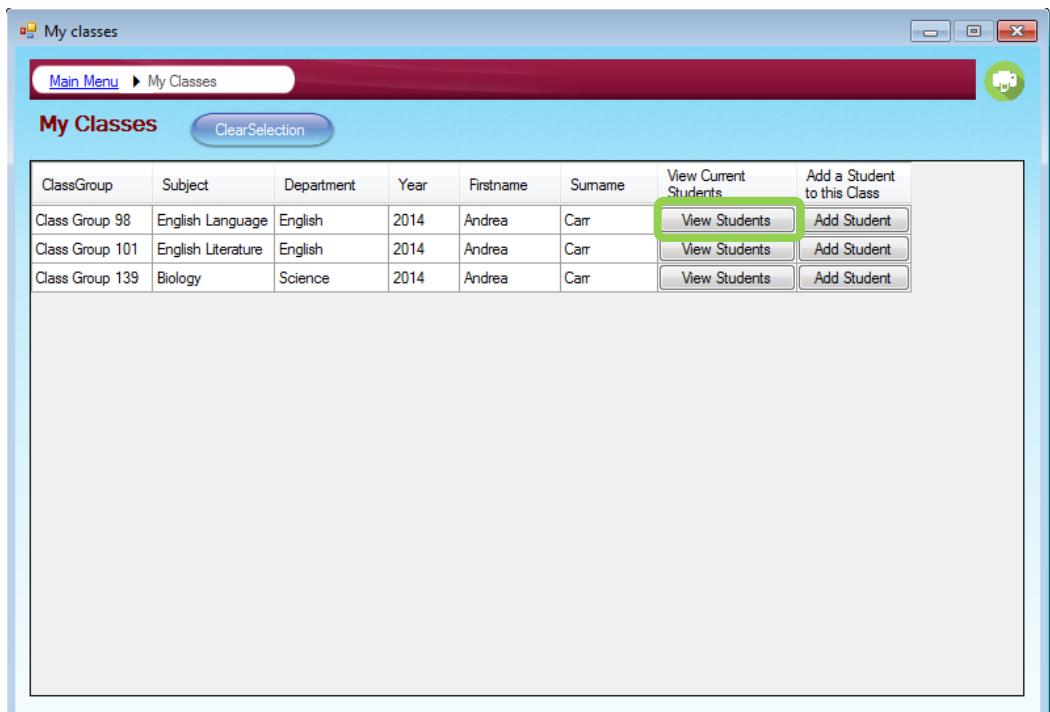
Reference 35

Evidence showing the attempted deletion of a class.



Reference 36

Evidence to show the functionality of the data grid.



Main Menu ► My Classes

My Classes ClearSelection

Firstname	Surname	Form	ClassGroup	Subject	Department	Year	Remove this Student from this class
Abigail	Phoebe	12G	Class Group 98	English Language	English	2014	Remove Student
Abigail	Sophie	12A	Class Group 98	English Language	English	2014	Remove Student
Amelia-Jane	Georgia	12N	Class Group 98	English Language	English	2014	Remove Student
Amy	Adam	12M	Class Group 98	English Language	English	2014	Remove Student
Brogan	Arnold	12B	Class Group 98	English Language	English	2014	Remove Student
Caitlin	Hannah	12J	Class Group 98	English Language	English	2014	Remove Student
Charlotte	Joe	12P	Class Group 98	English Language	English	2014	Remove Student
Emily	Joe	12A	Class Group 98	English Language	English	2014	Remove Student
Emily	Omari	12P	Class Group 98	English Language	English	2014	Remove Student
Francesca	Charlotte	12W	Class Group 98	English Language	English	2014	Remove Student
Gemma	Lizzie	12B	Class Group 98	English Language	English	2014	Remove Student
Georgia	Mabel	12N	Class Group 98	English Language	English	2014	Remove Student
Georgie	Ben	12B	Class Group 98	English Language	English	2014	Remove Student
Hannah	Tom	12B	Class Group 98	English Language	English	2014	Remove Student
Holly	Ben	12M	Class Group 98	English Language	English	2014	Remove Student
Isabella	Amy	12J	Class Group 98	English Language	English	2014	Remove Student
Karen	Umarah	12A	Class Group 98	English Language	English	2014	Remove Student

Reference 37

Evidence to show the functionality of the data grid.

The image shows two windows from a Windows application. The top window is titled "My classes" and displays a data grid titled "My Classes". The grid has columns: ClassGroup, Subject, Department, Year, Firstname, Surname, View Current Students, and Add a Student to this Class. There are three rows of data:

ClassGroup	Subject	Department	Year	Firstname	Surname	View Current Students	Add a Student to this Class
Class Group 98	English Language	English	2014	Andrea	Car	View Students	Add Student
Class Group 101	English Literature	English	2014	Andrea	Car	View Students	Add Student
Class Group 139	Biology	Science	2014	Andrea	Car	View Students	Add Student

A green rounded rectangle highlights the "Add Student" button in the last row. A large blue arrow points downwards from the bottom right corner of the "My classes" window to the bottom window. The bottom window is titled "Add Student Class" and shows a sub-menu path: Main Menu > Student Classes > Add Student. The main title is "Add Student to Class". It displays the "Current Selected Class:" as "Class Group 98 English Language 2014". Below this is a search field labeled "Search for a Student" and a dropdown menu labeled "Select an attribute:".

Reference 38

Evidence to show the functionality of the data grid.

The image shows a Windows application interface with two windows. The top window is titled 'My classes' and displays a data grid of student records. The bottom window is titled 'Remove Student Class' and is a dialog box for removing a student from a class.

My Classes Window Data:

Firstname	Surname	Form	ClassGroup	Subject	Department	Year	Remove this Student from this class
Abigail	Phoebe	12G	Class Group 98	English Language	English	2014	Remove Student
Abigail	Sophie	12A	Class Group 98	English Language	English	2014	Remove Student
Amelia-Jane	Georgia	12N	Class Group 98	English Language	English	2014	Remove Student
Amy	Adam	12M	Class Group 98	English Language	English	2014	Remove Student
Brogan	Arnold	12B	Class Group 98	English Language	English	2014	Remove Student
Caitlin	Hannah	12J	Class Group 98	English Language	English	2014	Remove Student
Charlotte	Joe	12P	Class Group 98	English Language	English	2014	Remove Student
Emily	Joe	12A	Class Group 98	English Language	English	2014	Remove Student
Emily	Omari	12P	Class Group 98	English Language	English	2014	Remove Student
Francesca	Charlotte	12W	Class Group 98	English Language	English	2014	Remove Student
Gemma	Lizzie	12B	Class Group 98	English Language	English	2014	Remove Student
Georgia	Mabel	12N	Class Group 98	English Language	English	2014	Remove Student
Georgie	Ben	12B	Class Group 98	English Language	English	2014	Remove Student
Hannah	Tom	12B	Class Group 98	English Language	English	2014	Remove Student
Holly	Ben	12M	Class Group 98	English Language	English	2014	Remove Student
Isabella	Amy	12J	Class Group 98	English Language	English	2014	Remove Student
Total	Total	12A	Class Group 98	English Language	English	2014	Remove Student

Remove Student Class Window Data:

ClassGroup	Subject	Year
Class Group 98	English Language	2014

Student Firstname: Abigail
Student Surname: Phoebe
Student Form: 12G

Remove from Class

Reference 39

Evidence to show the functionality of the clear selection button.

The screenshot displays two windows of a software application titled "My classes".

Top Window: This window shows a grid of student records. The columns are labeled: Firstname, Surname, Form, ClassGroup, Subject, Department, Year, and "Remove this Student from this class". A green oval highlights the "ClearSelection" button at the top right of the grid area. The data in the grid is as follows:

Firstname	Surname	Form	ClassGroup	Subject	Department	Year	Remove this Student from this class
Abigail	Phoebe	12G	Class Group 98	English Language	English	2014	Remove Student
Abigail	Sophie	12A	Class Group 98	English Language	English	2014	Remove Student
Amelia-Jane	Georgia	12N	Class Group 98	English Language	English	2014	Remove Student
Amy	Adam	12M	Class Group 98	English Language	English	2014	Remove Student
Brogan	Arnold	12B	Class Group 98	English Language	English	2014	Remove Student
Caitlin	Hannah	12J	Class Group 98	English Language	English	2014	Remove Student
Charlotte	Joe	12P	Class Group 98	English Language	English	2014	Remove Student
Emily	Joe	12A	Class Group 98	English Language	English	2014	Remove Student
Emily	Omari	12P	Class Group 98	English Language	English	2014	Remove Student
Francesca	Charlotte	12W	Class Group 98	English Language	English	2014	Remove Student
Gemma	Lizzie	12B	Class Group 98	English Language	English	2014	Remove Student
Georgia	Mabel	12N	Class Group 98	English Language	English	2014	Remove Student
Georgie	Ben	12B	Class Group 98	English Language	English	2014	Remove Student
Hannah	Tom	12B	Class Group 98	English Language	English	2014	Remove Student
Holly	Ben	12M	Class Group 98	English Language	English	2014	Remove Student
Isabella	Amy	12J	Class Group 98	English Language	English	2014	Remove Student
			Class Group 99	English Language	English	2014	Remove Student

Bottom Window: This window shows the same grid, but the data has been cleared. Only the first row of the grid remains, indicating that all previous data has been removed. The "ClearSelection" button is now highlighted with a green oval.

Reference 40

Evidence to show the functionality of the data grid.

The screenshot shows two instances of a Windows application window titled "My Class Grades".
The top window displays a data grid with the following columns: ClassGroup, Subject, Department, Year, Firstname, Surname, and three "View Grades" buttons. The "View Grades" buttons in the second and third rows are also highlighted with a green box.
The bottom window shows a larger data grid with more detailed student information: Firstname, Surname, Subject, Grade, Level, GradeType, Year, Edit Class Grade, and View all Student's Grades. This grid lists numerous students, each with their first name, last name, subject (English Language), grade (A, B, C), level (KS4), grade type (Achieved), year (2014), edit grade button, and view grades button.
A large blue arrow points downwards from the top window to the bottom one, indicating a transition or comparison between the two views.

ClassGroup	Subject	Department	Year	Firstname	Surname	View Grades
Class Group 98	English Language	English	2014	Andrea	Carr	View Grades
Class Group 101	English Literature	English	2014	Andrea	Carr	View Grades
Class Group 139	Biology	Science	2014	Andrea	Carr	View Grades

Firstname	Surname	Subject	Grade	Level	GradeType	Year	Edit Class Grade	View all Student's Grades
Abigail	Phoebe	English Language	A	KS4	Achieved	2014	Edit Grade	View Grades
Abigail	Sophie	English Language	A	KS4	Achieved	2014	Edit Grade	View Grades
Amelia-Jane	Georgia	English Language	A	KS4	Achieved	2014	Edit Grade	View Grades
Amy	Adam	English Language	A	KS4	Achieved	2014	Edit Grade	View Grades
Brogan	Arnold	English Language	B	KS4	Achieved	2014	Edit Grade	View Grades
Caitlin	Hannah	English Language	B	KS4	Achieved	2014	Edit Grade	View Grades
Charlotte	Joe	English Language	B	KS4	Achieved	2014	Edit Grade	View Grades
Emily	Joe	English Language	C	KS4	Achieved	2014	Edit Grade	View Grades
Emily	Omari	English Language	A	KS4	Achieved	2014	Edit Grade	View Grades
Francesca	Charlotte	English Language	B	KS4	Achieved	2014	Edit Grade	View Grades
Gemma	Lizzie	English Language	A	KS4	Achieved	2014	Edit Grade	View Grades
Georgia	Mabel	English Language	B	KS4	Achieved	2014	Edit Grade	View Grades
Georgie	Ben	English Language	A	KS4	Achieved	2014	Edit Grade	View Grades
Hannah	Tom	English Language	A*	KS4	Achieved	2014	Edit Grade	View Grades
Holly	Ben	English Language	B	KS4	Achieved	2014	Edit Grade	View Grades
Isabella	Amy	English Language	B	KS4	Achieved	2014	Edit Grade	View Grades
Isobel	Hannah	English Language	A	KS4	Achieved	2014	Edit Grade	View Grades

Reference 41

Evidence to show the functionality of the data grid.

The screenshot shows a Windows application window titled "My Class Grades". The main area contains a data grid with columns: Firstname, Surname, Subject, Grade, Level, GradeType, Year, Edit Class Grade, and View all Student's Grades. The "Edit Class Grade" button for the first row is highlighted with a green border. The "View all Student's Grades" button for the same row is also visible. Below the grid, there is a "ClearSelection" button. The "Edit Class Grade" button is located at approximately [165, 600, 185, 750]. The "View all Student's Grades" button is located at approximately [165, 760, 185, 810].

Edit Grade

Main Menu > Grade > Edit Grade

Edit Grade

Firstname:	Abigail
Surname:	Phoebe
Subject:	English Language
Level:	KS4
Grade:	A
Grade Type:	Achieved
Year:	2014

Buttons: Edit Grade, Delete

Reference 42

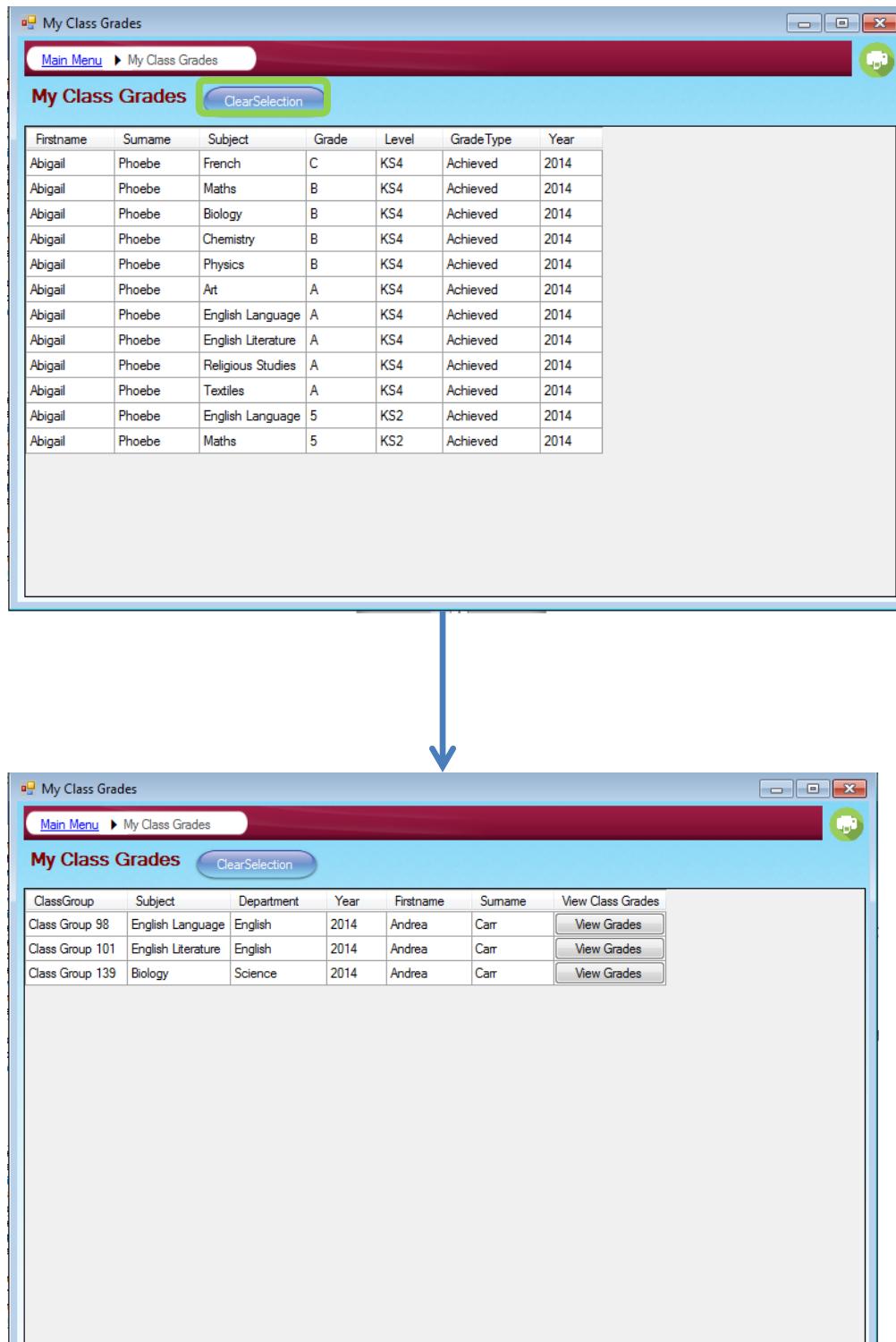
Evidence to show the functionality of the data grid.

Firstname	Surname	Subject	Grade	Level	GradeType	Year	Edit Class Grade	View all Student's Grades
Abigail	Phoebe	English Language	A	KS4	Achieved	2014	Edit Grade	View Grades
Abigail	Sophie	English Language	A	KS4	Achieved	2014	Edit Grade	View Grades
Amelia-Jane	Georgia	English Language	A	KS4	Achieved	2014	Edit Grade	View Grades
Amy	Adam	English Language	A	KS4	Achieved	2014	Edit Grade	View Grades
Brogan	Arnold	English Language	B	KS4	Achieved	2014	Edit Grade	View Grades
Caitlin	Hannah	English Language	B	KS4	Achieved	2014	Edit Grade	View Grades
Charlotte	Joe	English Language	B	KS4	Achieved	2014	Edit Grade	View Grades
Emily	Joe	English Language	C	KS4	Achieved	2014	Edit Grade	View Grades
Emily	Omari	English Language	A	KS4	Achieved	2014	Edit Grade	View Grades
Francesca	Charlotte	English Language	B	KS4	Achieved	2014	Edit Grade	View Grades
Gemma	Lizzie	English Language	A	KS4	Achieved	2014	Edit Grade	View Grades
Georgia	Mabel	English Language	B	KS4	Achieved	2014	Edit Grade	View Grades
Georgie	Ben	English Language	A	KS4	Achieved	2014	Edit Grade	View Grades
Hannah	Tom	English Language	A*	KS4	Achieved	2014	Edit Grade	View Grades
Holly	Ben	English Language	B	KS4	Achieved	2014	Edit Grade	View Grades
Isabella	Amy	English Language	B	KS4	Achieved	2014	Edit Grade	View Grades
Isobel	Hannah	English Language	A	KS4	Achieved	2014	Edit Grade	View Grades

Firstname	Surname	Subject	Grade	Level	GradeType	Year
Abigail	Phoebe	French	C	KS4	Achieved	2014
Abigail	Phoebe	Maths	B	KS4	Achieved	2014
Abigail	Phoebe	Biology	B	KS4	Achieved	2014
Abigail	Phoebe	Chemistry	B	KS4	Achieved	2014
Abigail	Phoebe	Physics	B	KS4	Achieved	2014
Abigail	Phoebe	Art	A	KS4	Achieved	2014
Abigail	Phoebe	English Language	A	KS4	Achieved	2014
Abigail	Phoebe	English Literature	A	KS4	Achieved	2014
Abigail	Phoebe	Religious Studies	A	KS4	Achieved	2014
Abigail	Phoebe	Textiles	A	KS4	Achieved	2014
Abigail	Phoebe	English Language	5	KS2	Achieved	2014
Abigail	Phoebe	Maths	5	KS2	Achieved	2014

Reference 43

Evidence to show the functionality of the clear selection button.



Reference 44

Test evidence which shows filtering by a form.

Grade Predictions

Main Menu > Grade Predictions

Predicted Grades

View Individual Student's A2 Predictions

Select an attribute: Form

Form: 13W Name Gender Year

Search

Or

View a Form's A2 Predicted Grades

Select Academic Year:

Or

View a Class's A2 Predicted Grades

Select a Subject:

Select Academic Year:

Or

View a Year's A2 Predicted Grades

Select Academic Year:

ClearSelection

Grade Predictions

Main Menu > Grade Predictions

Predicted Grades

View Individual Student's A2 Predictions

Select an attribute: Form

Form: 13W

Search

Or

View a Form's A2 Predicted Grades

Select Academic Year:

Or

View a Class's A2 Predicted Grades

Select a Subject:

Select Academic Year:

Or

View a Year's A2 Predicted Grades

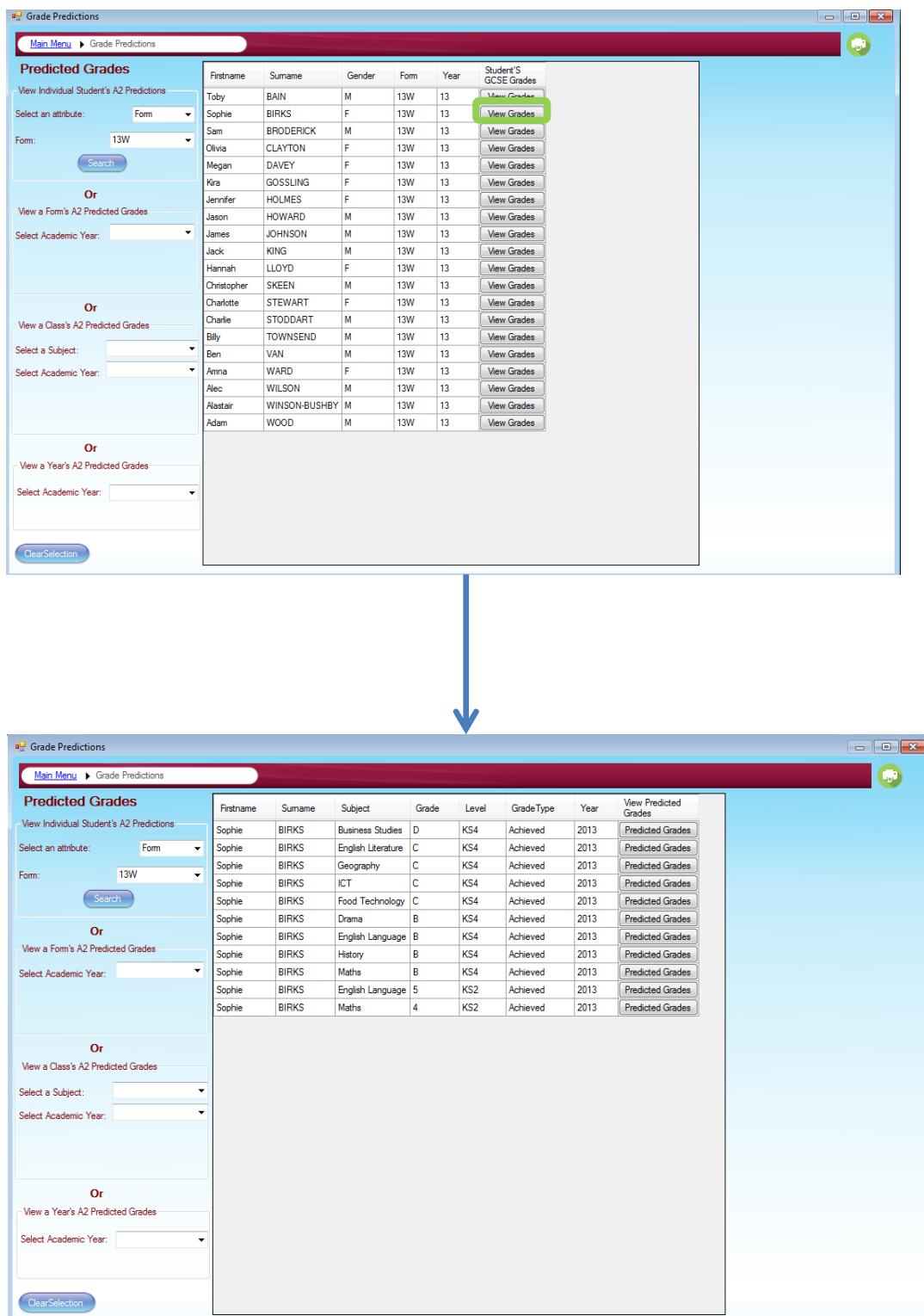
Select Academic Year:

ClearSelection

Firstname	Surname	Gender	Form	Year	Student'S GCSE Grades
Toby	BAIN	M	13W	13	View Grades
Sophie	BIRKS	F	13W	13	View Grades
Sam	BRODERICK	M	13W	13	View Grades
Olivia	CLAYTON	F	13W	13	View Grades
Megan	DAVEY	F	13W	13	View Grades
Kira	GOSSLING	F	13W	13	View Grades
Jennifer	HOLMES	F	13W	13	View Grades
Jason	HOWARD	M	13W	13	View Grades
James	JOHNSON	M	13W	13	View Grades
Jack	KING	M	13W	13	View Grades
Hannah	LLOYD	F	13W	13	View Grades
Christopher	SKEEN	M	13W	13	View Grades
Charlotte	STEWART	F	13W	13	View Grades
Charlie	STODDART	M	13W	13	View Grades
Billy	TOWNSEND	M	13W	13	View Grades
Ben	VAN	M	13W	13	View Grades
Anna	WARD	F	13W	13	View Grades
Alec	WILSON	M	13W	13	View Grades
Alastair	WINSON-BUSHBY	M	13W	13	View Grades
Adam	WOOD	M	13W	13	View Grades

Reference 45

Test evidence which shows the functionality of the data grid.



The screenshot displays two instances of a Windows application window titled "Grade Predictions".

Top Window (Initial View):

- Predicted Grades:** This section contains three search options:
 - View Individual Student's A2 Predictions:** Form dropdown set to "13W", Search button.
 - Or**
 - View a Form's A2 Predicted Grades:** Select Academic Year dropdown.
- Student's GCSE Grades:** A table listing student names, surnames, genders, forms, years, and a "View Grades" button for each row.

Firstname	Surname	Gender	Form	Year	Student's GCSE Grades
Toby	BAIN	M	13W	13	View Grades
Sophie	BIRKS	F	13W	13	View Grades
Sam	BRODERICK	M	13W	13	View Grades
Olivia	CLAYTON	F	13W	13	View Grades
Megan	DAVEY	F	13W	13	View Grades
Kira	GOSSLING	F	13W	13	View Grades
Jennifer	HOLMES	F	13W	13	View Grades
Jason	HOWARD	M	13W	13	View Grades
James	JOHNSON	M	13W	13	View Grades
Jack	KING	M	13W	13	View Grades
Hannah	LLOYD	F	13W	13	View Grades
Christopher	SKEEN	M	13W	13	View Grades
Charlotte	STEWART	F	13W	13	View Grades
Charlie	STODDART	M	13W	13	View Grades
Billy	TOWNSEND	M	13W	13	View Grades
Ben	VAN	M	13W	13	View Grades
Anna	WARD	F	13W	13	View Grades
Alec	WILSON	M	13W	13	View Grades
Nastair	WINSON-BUSHBY	M	13W	13	View Grades
Adam	WOOD	M	13W	13	View Grades

Bottom Window (After Selection):

- Predicted Grades:** Same search options as the top window.
- Predicted Grades:** A table listing predicted grades for Sophie BIRKS across various subjects and levels.

Firstname	Surname	Subject	Grade	Level	GradeType	Year	View Predicted Grades
Sophie	BIRKS	Business Studies	D	KS4	Achieved	2013	Predicted Grades
Sophie	BIRKS	English Literature	C	KS4	Achieved	2013	Predicted Grades
Sophie	BIRKS	Geography	C	KS4	Achieved	2013	Predicted Grades
Sophie	BIRKS	ICT	C	KS4	Achieved	2013	Predicted Grades
Sophie	BIRKS	Food Technology	C	KS4	Achieved	2013	Predicted Grades
Sophie	BIRKS	Drama	B	KS4	Achieved	2013	Predicted Grades
Sophie	BIRKS	English Language	B	KS4	Achieved	2013	Predicted Grades
Sophie	BIRKS	History	B	KS4	Achieved	2013	Predicted Grades
Sophie	BIRKS	Maths	B	KS4	Achieved	2013	Predicted Grades
Sophie	BIRKS	English Language	5	KS2	Achieved	2013	Predicted Grades
Sophie	BIRKS	Maths	4	KS2	Achieved	2013	Predicted Grades

Reference 46

Test evidence which shows the functionality of the data grid.

The screenshot shows a Windows application window titled "Grade Predictions". On the left, there is a sidebar with several search and selection options:

- "View Individual Student's A2 Predictions": "Select an attribute: Form", "Form: 13W", "Search" button.
- "Or": "View a Form's A2 Predicted Grades": "Select Academic Year: [dropdown]".
- "Or": "View a Class's A2 Predicted Grades": "Select a Subject: [dropdown]", "Select Academic Year: [dropdown]".
- "Or": "View a Year's A2 Predicted Grades": "Select Academic Year: [dropdown]".
- "Clear Selection" button.

The main area contains a data grid table with the following columns: Firstname, Surname, Subject, Grade, Level, GradeType, Year, and "View Predicted Grades". The rows show data for Sophie BIRKS across different subjects and years:

Firstname	Surname	Subject	Grade	Level	GradeType	Year	View Predicted Grades
Sophie	BIRKS	Business Studies	D	KS4	Achieved	2013	Predicted Grade
Sophie	BIRKS	English Literature	C	KS4	Achieved	2013	Predicted Grade
Sophie	BIRKS	Geography	C	KS4	Achieved	2013	Predicted Grade
Sophie	BIRKS	ICT	C	KS4	Achieved	2013	Predicted Grade
Sophie	BIRKS	Food Technology	C	KS4	Achieved	2013	Predicted Grade
Sophie	BIRKS	Drama	B	KS4	Achieved	2013	Predicted Grade
Sophie	BIRKS	English Language	B	KS4	Achieved	2013	Predicted Grade
Sophie	BIRKS	History	B	KS4	Achieved	2013	Predicted Grade
Sophie	BIRKS	Maths	B	KS4	Achieved	2013	Predicted Grade
Sophie	BIRKS	English Language	5	KS2	Achieved	2013	Predicted Grade
Sophie	BIRKS	Maths	4	KS2	Achieved	2013	Predicted Grade



This screenshot is identical to the one above, showing the "Grade Predictions" application interface. The sidebar and data grid are the same. A secondary data grid is overlaid on the right side of the main grid, titled "Predicted A2". It has two columns: "Subject" and "Predicted A2 Grade". The data is as follows:

Subject	Predicted A2 Grade
Art	C
Biology	D
Business Studies	C
Chemistry	D
Computing	D
Drama	C
Economics	C
English Language	D
English Literature	D
Food Technology	C
French	D
Geography	D
German	D
Graphics	C
Health And Social Care	C
History	D
ICT	D
Leisure and Tourism	C
Maths	D
Music	C
Physical Education	D
Physics	D
Religious Studies	C
Resistant Materials	D
Spanish	D
Textiles	C

Reference 47

Evidence which shows the generation of predicted grades for a form group.

The image shows two screenshots of a Windows application titled "Grade Predictions".

Screenshot 1 (Top): This screenshot shows the search interface. It has three main sections:

- View Individual Student's A2 Predictions:** A dropdown menu labeled "Select an attribute".
- Or:** A section labeled "View a Form's A2 Predicted Grades" with dropdown menus for "Select Academic Year" (set to 2015) and "Select a form" (set to 11K), followed by a "Search" button.
- Or:** A section labeled "View a Class's A2 Predicted Grades" with dropdown menus for "Select a Subject" and "Select Academic Year".
- Or:** A section labeled "View a Year's A2 Predicted Grades" with a dropdown menu for "Select Academic Year".

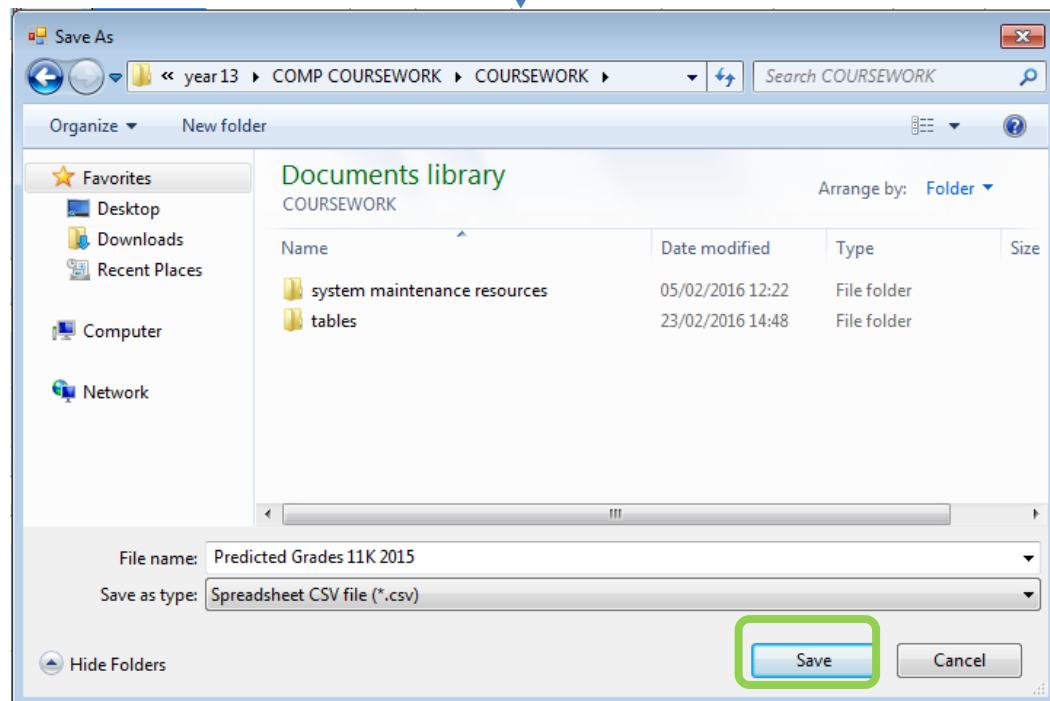
Screenshot 2 (Bottom): This screenshot shows the results of the search. It displays a table of predicted grades for students in form 11K. The columns represent various subjects: Firstname, Surname, Art, Biology, Business Studies, Chemistry, Computing, Drama, Economics, English Language, and English Literature. The rows list individual students with their predicted grades in each subject. A large blue arrow points from the top screenshot down to the bottom one, indicating the flow from search input to the generated results table.

Firstname	Surname	Art	Biology	Business Studies	Chemistry	Computing	Drama	Economics	English Language	English Literature
Ela	WATSON	A	C	A	C	B	B	A	B	B
Poppy	WATSON	A	C	B	C	C	B	B	B	B
Dulcie	UPCHURCH	A	C	A	C	B	B	A	B	B
Lauren	TOMPKINSON	B	C	B	C	C	B	B	B	B
Colette	THOMPSON	C	D	C	D	D	C	C	C	C
Dan	TAYLOR	B	C	B	D	C	B	B	C	C
Fearn	SPENDLOVE	A	C	A	C	B	A	A	B	B
James	PATON	B	D	B	D	C	C	B	C	C
Reed	PAISH	B	C	B	C	C	B	B	B	B
Euan	JOHNSON-EVANS	A	B	A	B	B	A	A	A	A
Zoe	GRIFFITHS	A	C	A	C	B	B	A	B	B
Gabriel	DEAVILLE	C	D	C	D	D	C	C	C	C
Joseph	DAVIES	B	C	B	D	C	B	B	C	C
Martin	DARBYSHIRE	C	D	C	D	D	D	C	D	D
Freddie	CARTER	B	D	B	D	C	C	B	C	C
Jack	CAIN	B	C	B	C	C	B	B	B	B
Freya	BRAMBILLA	B	D	B	D	C	C	B	C	C
Chris	BARR	B	D	C	D	C	C	C	C	C

Reference 48

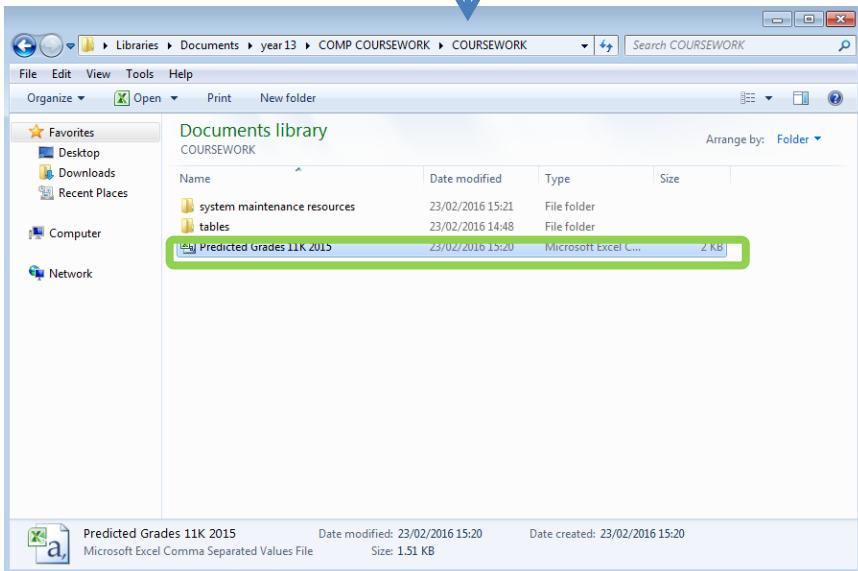
Evidence which shows how a CSV spread sheet file of grades is generated.

Firstname	Surname	Art	Biology	Business Studies	Chemistry	Computing	Drama	Economics	English Language	English Literature
Ella	WATSON	A	C	A	C	B	B	A	B	B
Poppy	WATSON	A	C	B	C	B	B	B	B	B
Dulcie	UPCHURCH	A	C	A	C	B	B	A	B	B
Lauren	TOMPKINSON	B	C	B	C	C	B	B	B	B
Colette	THOMPSON	C	D	C	D	D	C	C	C	C
Dan	TAYLOR	B	C	B	D	C	B	B	C	C
Fearn	SPENDLOVE	A	C	A	C	B	A	A	B	B
James	PATON	B	D	B	D	C	C	B	C	C
Reed	PAISH	B	C	B	C	C	B	B	B	B
Euan	JOHNSON-EVANS	A	B	A	B	B	A	A	A	A
Zoe	GRIFFITHS	A	C	A	C	B	B	A	B	B
Gabriel	DEAVILLE	C	D	C	D	D	C	C	C	C
Joseph	DAVIES	B	C	B	D	C	B	B	C	C
Martin	DARBYSHIRE	C	D	C	D	D	D	C	D	D
Freddie	CARTER	B	D	B	D	C	C	B	C	C
Jack	CAIN	B	C	B	C	C	B	B	B	B
Freya	BRAMBILLA	B	D	B	D	C	C	B	C	C
Chris	BARR	B	D	C	D	C	C	C	C	C

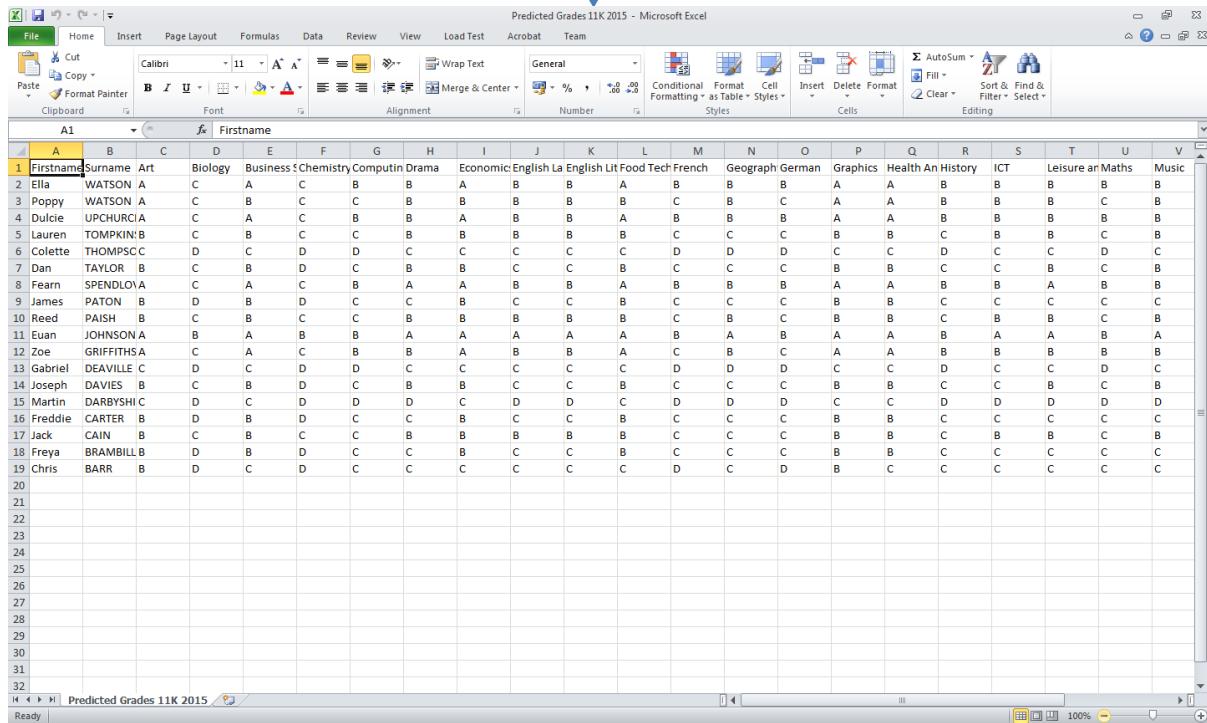


Reference 48 (continued)

↓



↓



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V				
1	Firstnames	Surname	Art	Biology	Business	Chemistry	Computin	Drama	Economic	English	La	English	Lit	Food	Tech	French	Geograph	German	Graphics	Health	An History	ICT	Leisure	air	Maths	Music
2	Ella	WATSON	A	C	A	C	B	B	A	B	B	A	B	B	B	A	A	A	B	B	B	B	B	B	B	
3	Poppy	WATSON	A	C	B	C	B	B	B	B	B	B	B	B	C	B	A	A	B	B	B	B	C	B		
4	Dulcie	UPCHURCIA	C	A	C	B	B	A	B	B	A	B	B	B	B	A	A	A	B	B	B	B	B	B		
5	Lauren	TOMPKIN	B	C	B	C	B	B	B	B	B	B	B	C	C	C	B	B	B	C	B	B	C	B		
6	Colette	THOMPSC	D	C	D	D	C	C	C	C	C	C	D	D	D	C	C	D	C	C	C	C	D	C		
7	Dan	TAYLOR	B	C	B	D	C	B	B	C	C	B	C	C	C	B	B	C	B	C	C	C	B	C		
8	Fearn	SPENDLOA	C	A	C	B	A	A	B	B	A	B	B	B	B	A	A	B	B	A	B	A	B	B		
9	James	PATON	B	D	B	D	C	C	B	C	B	C	C	C	B	B	B	C	C	C	C	C	C			
10	Reed	PAISH	B	C	B	C	C	B	B	B	B	B	B	B	C	B	B	C	B	B	C	B	C			
11	Euan	JOHNSON	A	B	B	B	A	A	A	A	B	A	B	A	B	A	A	B	A	A	B	A	A			
12	Zoe	GRIFITHS	A	C	A	B	B	A	B	B	A	C	B	C	A	A	B	B	B	B	B	B	B			
13	Gabriel	DEAVILLE	C	D	C	D	C	C	C	C	D	D	D	D	C	C	D	C	C	C	D	C				
14	Joseph	DAVIES	B	C	B	D	C	B	C	C	C	C	C	C	B	B	C	C	B	C	C	B				
15	Martin	DARBYSHI	C	D	C	D	D	C	D	D	C	D	D	D	C	C	D	D	D	D	D					
16	Freddie	CARTER	B	D	B	D	C	C	B	C	C	B	C	C	B	B	C	C	C	C	C					
17	Jack	CAIN	B	C	B	C	C	B	B	B	B	C	C	C	B	B	C	B	B	C	B					
18	Freya	BRAMBILL	B	D	B	D	C	C	B	C	C	B	C	C	B	B	C	C	C	C	C					
19	Chris	BARR	B	D	C	D	C	C	C	C	D	C	D	B	C	C	C	C	C	C						
20																										
21																										
22																										
23																										
24																										
25																										
26																										
27																										
28																										
29																										
30																										
31																										
32																										

Reference 49

Test evidence which shows the functionality of the data grid.

The screenshot displays a Windows application interface for managing student grades. It consists of two main windows: a larger 'Grade' window at the top and a smaller 'Edit Grade' window at the bottom, connected by a large blue downward-pointing arrow.

Grade Window:

- Title Bar:** Grade
- Menu Bar:** Main Menu > Grade
- Toolbar:** Includes icons for Back, Forward, Home, and Help.
- Form:** 'View/Edit a students grade' with fields for Firstname, Surname, Subject, Grade, Level, GradeType, Year, and a 'Grade Details' button.
- Data Grid:** A table listing student grades with columns: Firstname, Surname, Subject, Grade, Level, GradeType, Year, and Grade Details (with the last column being a button).
- Buttons:** Search, ClearSelection, and Add a Grade or view a Student's Grades.

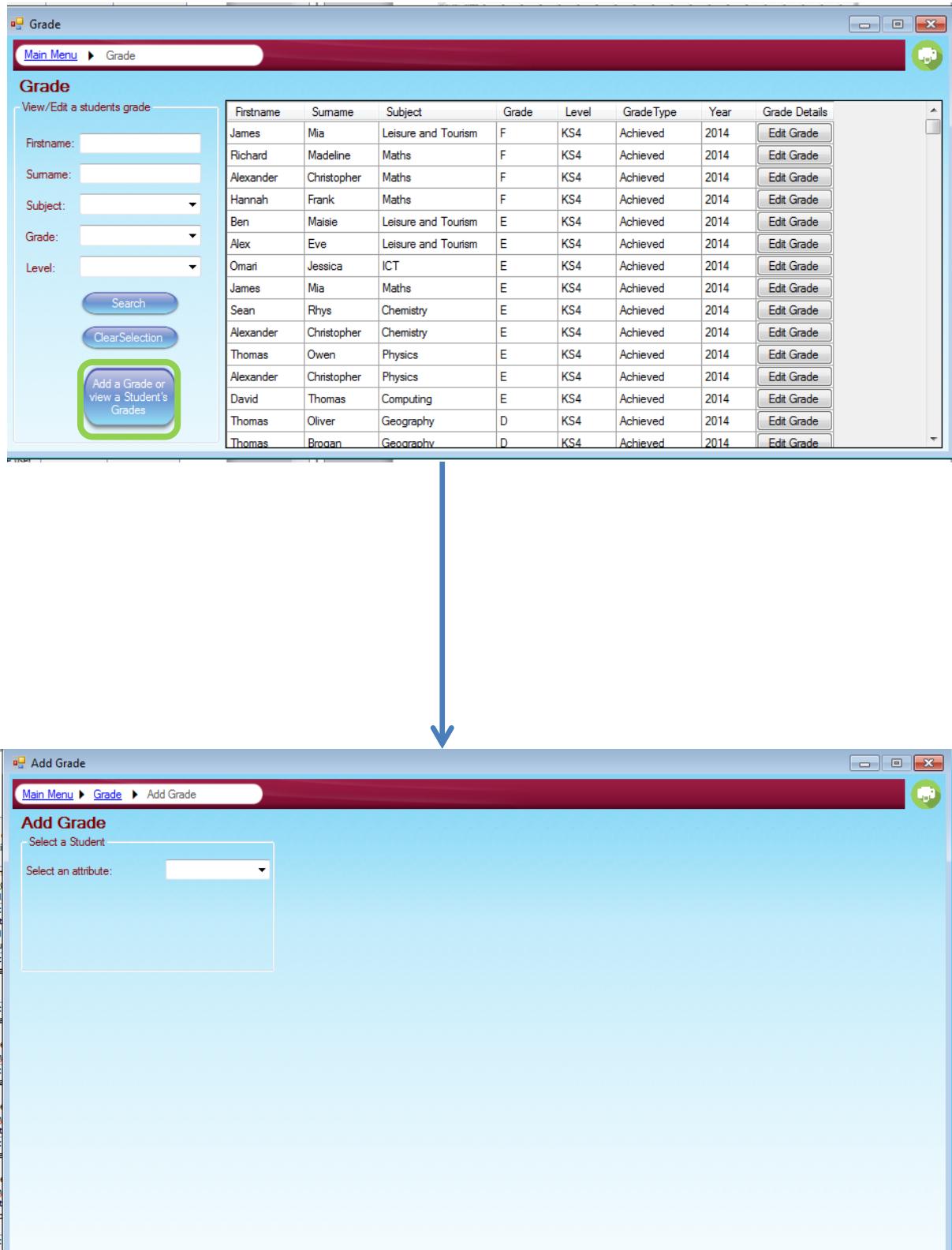
Edit Grade Window:

- Title Bar:** Edit Grade
- Menu Bar:** Main Menu > Grade > Edit Grade
- Form:** 'Edit Grade' with fields for Firstname, Surname, Subject, Level, Grade, Grade Type, and Year.
- Buttons:** Edit Grade and Delete.

A large blue arrow points from the Grade window down to the Edit Grade window, indicating a flow or relationship between the two screens.

Reference 50

Evidence to show the test of clicking the add a grade button.



Reference 51

Test evidence which shows that cumulative filters on the Grade form function correctly so that an admin user can filter any grade in the system.

The screenshot shows the 'Grade' application window. At the top left is the 'Main Menu' with a 'Grade' option. The main area is titled 'Grade' and contains a form for 'View/Edit a student's grade'. The form includes fields for 'Firstname' (joe), 'Surname' (call), 'Subject' (French), 'Grade' (C), and 'Level' (KS4). Below the form are two buttons: 'Search' and 'ClearSelection'. A large green rounded rectangle highlights the entire search form area. At the bottom right of the form is a blue button labeled 'Add a Grade or view a Student's Grades'.



The screenshot shows the 'Grade' application window after a search has been performed. The search form at the top left remains the same as in the previous screenshot. The main area now displays a table of student grades. The table has columns: Firstname, Surname, Subject, Grade, Level, GradeType, Year, and Grade Details. One row is highlighted with a green rounded rectangle, corresponding to the student 'Joe Callum' with a grade of 'C'. To the right of the table is a blue button labeled 'Edit Grade'.

Firstname	Surname	Subject	Grade	Level	GradeType	Year	Grade Details
Joe	Callum	French	C	KS4	Achieved	2014	Edit Grade

Reference 52

Test evidence which shows the functionality of the data grid.

Add Grade

Main Menu ▶ Grade ▶ Add Grade

Add Grade

Select a Student

Select an attribute: Form

Form: 12P

Search

Firstname	Surname	Gender	Form	Year	Grade Details
Owen	Joel	M	12P	12	View Grades
Omari	Jessica	M	12P	12	View Grades
Elliot	Emily	M	12P	12	View Grades
Joe	Callum	M	12P	12	View Grades
Tom	Sean	M	12P	12	View Grades
Natalie	Zak	F	12P	12	View Grades
Isobel	Daniel	F	12P	12	View Grades
Sanna	Abigail	F	12P	12	View Grades
Holly	Joe	F	12P	12	View Grades
Alice	Becky	F	12P	12	View Grades
Mimi	Oliver	F	12P	12	View Grades
Anna	Isobel	F	12P	12	View Grades
Viviana	Matthew	F	12P	12	View Grades
Isabel	Andrew	F	12P	12	View Grades
Alicia	George	F	12P	12	View Grades
Meredith	Lily	F	12P	12	View Grades
Charlotte	Joe	F	12P	12	View Grades
Emily	Omari	F	12P	12	View Grades



Add Grade

Main Menu ▶ Grade ▶ Add Grade

Add Grade

Select a Student

Select an attribute: Form

Form: 12P

Search

Add a Grade

Firstname: Owen

Surname: Joel

Subject:

Level:

Grade:

Grade Type: Achieved

Year Achieved:

Add Grade

Firstname	Surname	Subject	Grade	Level	GradeType	Year	Grade Details
Owen	Joel	Chemistry	C	KS4	Achieved	2014	Edit Grade
Owen	Joel	ICT	B	KS4	Achieved	2014	Edit Grade
Owen	Joel	Geography	A*	KS4	Achieved	2014	Edit Grade
Owen	Joel	Maths	A*	KS4	Achieved	2014	Edit Grade
Owen	Joel	English Language	A	KS4	Achieved	2014	Edit Grade
Owen	Joel	English Literature	A	KS4	Achieved	2014	Edit Grade
Owen	Joel	German	A	KS4	Achieved	2014	Edit Grade
Owen	Joel	Textiles	A	KS4	Achieved	2014	Edit Grade
Owen	Joel	Biology	A	KS4	Achieved	2014	Edit Grade
Owen	Joel	Physics	A	KS4	Achieved	2014	Edit Grade
Owen	Joel	English Language	4	KS2	Achieved	2014	Edit Grade
Owen	Joel	Maths	4	KS2	Achieved	2014	Edit Grade

Reference 53

Test evidence which shows the functionality of the data grid.

The image shows two windows of a Windows application. The top window is titled 'Add Grade' and the bottom window is titled 'Edit Grade'. A large blue arrow points from the 'Edit Grade' window back up to the 'Add Grade' window.

Add Grade Window:

- Main Menu: Main Menu > Grade > Add Grade
- Toolbar: Add Grade
- Left Panel: Select a Student
Select an attribute: Form
Form: 12P
Search
- Right Panel: Data Grid (Table 1)

Firstname	Surname	Subject	Grade	Level	GradeType	Year	Grade Details
Owen	Joel	Chemistry	C	KS4	Achieved	2014	Edit Grade
Owen	Joel	ICT	B	KS4	Achieved	2014	Edit Grade
Owen	Joel	Geography	A*	KS4	Achieved	2014	Edit Grade
Owen	Joel	Maths	A*	KS4	Achieved	2014	Edit Grade
Owen	Joel	English Language	A	KS4	Achieved	2014	Edit Grade
Owen	Joel	English Literature	A	KS4	Achieved	2014	Edit Grade
Owen	Joel	German	A	KS4	Achieved	2014	Edit Grade
Owen	Joel	Textiles	A	KS4	Achieved	2014	Edit Grade
Owen	Joel	Biology	A	KS4	Achieved	2014	Edit Grade
Owen	Joel	Physics	A	KS4	Achieved	2014	Edit Grade
Owen	Joel	English Language	4	KS2	Achieved	2014	Edit Grade
Owen	Joel	Maths	4	KS2	Achieved	2014	Edit Grade
- Buttons: Add Grade

Edit Grade Window:

- Main Menu: Main Menu > Grade > Edit Grade
- Toolbar: Edit Grade
- Form Fields:
 - Firstname: Owen
 - Surname: Joel
 - Subject: Geography
 - Level: KS4
 - Grade: A*
 - Grade Type: Achieved
 - Year: 2014
- Buttons: Edit Grade, Delete

Reference 54

Test evidence which shows the validation of the system to avoid data inconsistency.

The screenshot illustrates a validation process within a 'Add Grade' application. The interface consists of two main sections: a left panel for entering student details and a right panel for viewing a list of grades.

Left Panel (Add Grade):

- Select a Student:** A dropdown menu for 'Select an attribute' is set to 'Form', and the value '12P' is selected.
- Search:** A blue 'Search' button is present.
- Add a Grade:** A form with the following fields:
 - Firstname: Owen
 - Surname: Joel
 - Subject: (dropdown menu)
 - Level: KS4
 - Grade: (dropdown menu)
 - Grade Type: Achieved
 - Year Achieved: 2015
- Add Grade:** A blue 'Add Grade' button at the bottom of the form.

A green oval highlights the entire 'Add a Grade' section.

Right Panel:

Firstname	Surname	Subject	Grade	Level	GradeType	Year	Grade Details
Owen	Joel	Chemistry	C	KS4	Achieved	2014	<button>Edit Grade</button>
Owen	Joel	ICT	B	KS4	Achieved	2014	<button>Edit Grade</button>
Owen	Joel	Geography	A*	KS4	Achieved	2014	<button>Edit Grade</button>
Owen	Joel	Maths	A*	KS4	Achieved	2014	<button>Edit Grade</button>
Owen	Joel	English Language	A	KS4	Achieved	2014	<button>Edit Grade</button>
Owen	Joel	English Literature	A	KS4	Achieved	2014	<button>Edit Grade</button>
Owen	Joel	German	A	KS4	Achieved	2014	<button>Edit Grade</button>
Owen	Joel	Textiles	A	KS4	Achieved	2014	<button>Edit Grade</button>
Owen	Joel	Biology	A	KS4	Achieved	2014	<button>Edit Grade</button>
Owen	Joel	Physics	A	KS4	Achieved	2014	<button>Edit Grade</button>
Owen	Joel	English Language	4	KS2	Achieved	2014	<button>Edit Grade</button>
Owen	Joel	Maths	4	KS2	Achieved	2014	<button>Edit Grade</button>

A large blue arrow points downwards from the top application window to the second one.

Second Application Window:

The second window shows the same 'Add Grade' interface, but with validation errors. The 'Subject' field is highlighted in red, indicating it is required but empty. A modal dialog box is displayed, stating: "One or more textboxes have not been completed".

Reference 55

Test evidence which shows the validation of the system to avoid data inconsistency.

The screenshot shows the 'Add Grade' window. On the left, there's a 'Select a Student' section with dropdown menus for 'Select an attribute' (set to 'Form') and 'Form' (set to '12P'), followed by a 'Search' button. Below this is the 'Add a Grade' section, which contains fields for 'Firstname' (Owen), 'Surname' (Joel), 'Subject' (Music), 'Level' (KS4), 'Grade' (5), 'Grade Type' (Achieved), and 'Year Achieved' (2015). A green oval highlights the entire 'Add a Grade' section. At the bottom right of this section is a blue 'Add Grade' button, which is also highlighted with a green oval.



The screenshot shows the same 'Add Grade' window after the 'Add Grade' button was clicked. A modal dialog box has appeared in the center, containing an information icon and the text 'That grade cannot be applied to that Subject Level (Keystage)'. There is an 'OK' button at the bottom right of the dialog. The background of the main window is dimmed. A green oval highlights the modal dialog box.

Reference 56

Test evidence which shows what happens when a new grade is added to the system.

The screenshot shows the 'Add Grade' application window. On the left, there's a search interface for 'Select a Student' with dropdowns for 'Select an attribute' (set to 'Form'), 'Form' (set to '12P'), and a 'Search' button. Below it is the 'Add a Grade' form with fields for 'Firstname' (Owen), 'Surname' (Joel), 'Subject' (Drama), 'Level' (KS4), 'Grade' (D), 'Grade Type' (Achieved), and 'Year Achieved' (2015). A green rounded rectangle highlights this entire 'Add a Grade' section. To the right is a grid table showing student grades:

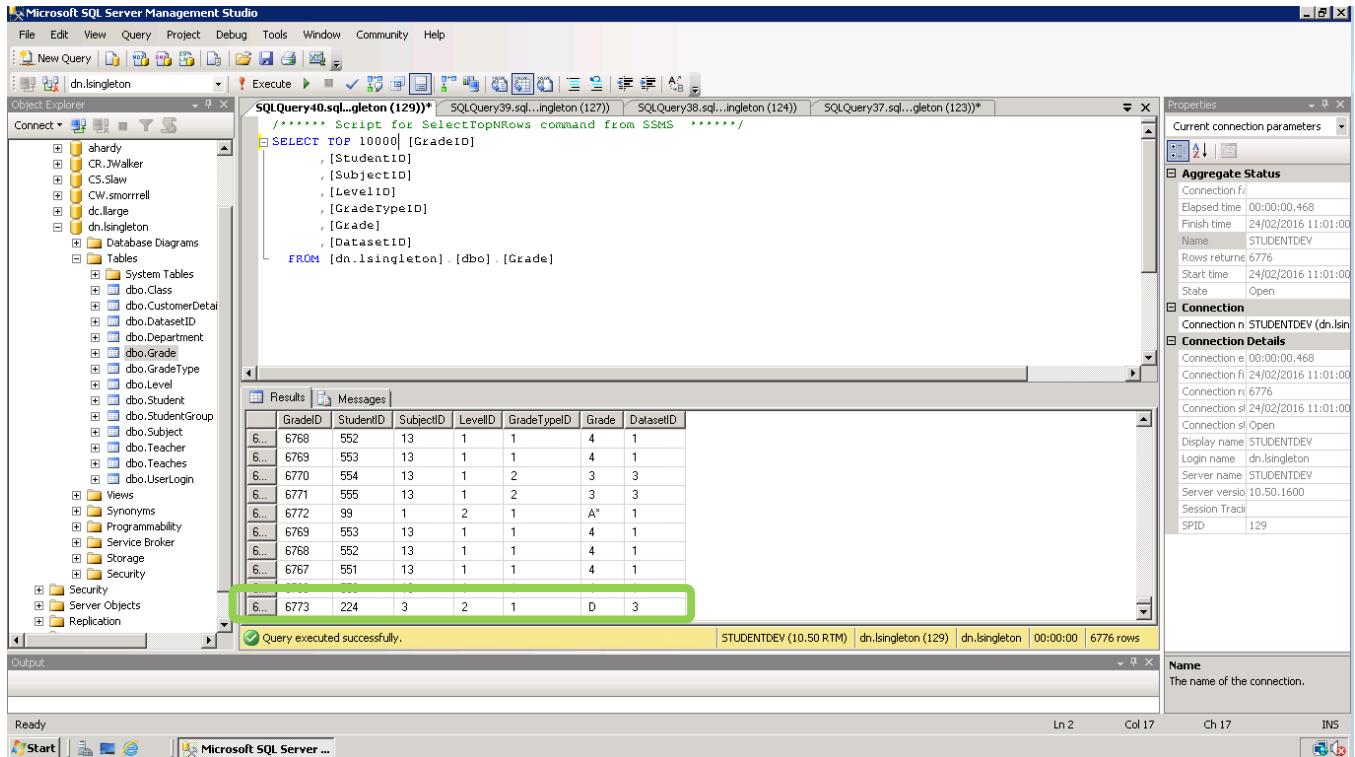
Firstname	Surname	Subject	Grade	Level	GradeType	Year	Grade Details
Owen	Joel	Chemistry	C	KS4	Achieved	2014	<button>Edit Grade</button>
Owen	Joel	ICT	B	KS4	Achieved	2014	<button>Edit Grade</button>
Owen	Joel	Geography	A*	KS4	Achieved	2014	<button>Edit Grade</button>
Owen	Joel	Maths	A*	KS4	Achieved	2014	<button>Edit Grade</button>
Owen	Joel	English Language	A	KS4	Achieved	2014	<button>Edit Grade</button>
Owen	Joel	English Literature	A	KS4	Achieved	2014	<button>Edit Grade</button>
Owen	Joel	German	A	KS4	Achieved	2014	<button>Edit Grade</button>
Owen	Joel	Textiles	A	KS4	Achieved	2014	<button>Edit Grade</button>
Owen	Joel	Biology	A	KS4	Achieved	2014	<button>Edit Grade</button>
Owen	Joel	Physics	A	KS4	Achieved	2014	<button>Edit Grade</button>
Owen	Joel	English Language	4	KS2	Achieved	2014	<button>Edit Grade</button>
Owen	Joel	Maths	4	KS2	Achieved	2014	<button>Edit Grade</button>



The screenshot shows the same 'Add Grade' application window after a new record was added. The 'Add a Grade' form now includes a 'Name' field with 'owen' typed into it. The 'Add Grade' button is highlighted with a green rounded rectangle. A modal dialog box titled 'Year11 Dynamic Grade data' with an information icon displays the message 'Record Added'. The 'OK' button of this dialog is also highlighted with a green rounded rectangle. The main grid table remains the same as in the first screenshot.

Reference 56 (continued)

The evidence below show the result of adding a new grade to the system and shows evidence of it being inserted into the SQL server database.



A screenshot of Microsoft SQL Server Management Studio (SSMS) showing a query results grid. The query executed successfully, returning 6776 rows. A new row has been highlighted with a green border, indicating a recent insertion. The columns in the results grid are GradeID, StudentID, SubjectID, LevelID, GradeTypeID, Grade, and DatasetID. The highlighted row contains values: GradeID 6773, StudentID 224, SubjectID 3, LevelID 2, GradeTypeID 1, Grade D, and DatasetID 3.

GradeID	StudentID	SubjectID	LevelID	GradeTypeID	Grade	DatasetID
6...	6768	552	13	1	4	1
6...	6769	553	13	1	4	1
6...	6770	554	13	1	2	3
6...	6771	555	13	1	2	3
6...	6772	99	1	2	1	A*
6...	6769	553	13	1	1	4
6...	6768	552	13	1	1	4
6...	6767	551	13	1	1	4
6...						
6773						
224						
3						
2						
1						
D						
3						

Reference 57

Test evidence which shows the validation of the system to avoid data inconsistency.

The screenshot illustrates a validation process within a Windows application titled 'Add Grade'. The application has a standard window title bar and a menu bar with 'Main Menu' > 'Grade' > 'Add Grade'. The main area is divided into two sections: 'Select a Student' on the left and 'Add a Grade' on the right.

Select a Student: A dropdown menu 'Select an attribute:' is set to 'Form', and the value '12P' is selected. A 'Search' button is present below the dropdown.

Add a Grade: This section contains fields for 'Firstname' (Owen), 'Surname' (Joel), 'Subject' (ICT), 'Level' (KS4), 'Grade' (D), 'Grade Type' (Achieved), and 'Year Achieved' (2015). A green oval highlights the entire 'Add a Grade' section, and a blue arrow points downwards from it towards the validation message.

Data Grid: To the right of the 'Add a Grade' section is a data grid displaying student grades. The columns are: Firstname, Surname, Subject, Grade, Level, GradeType, Year, and Grade Details. The data shows multiple entries for student 'Owen Joel' across various subjects like Chemistry, ICT, Geography, Maths, English Language, German, Textiles, Biology, Physics, and English Language at levels KS4 and KS2, with grades ranging from C to D and achievement years from 2014 to 2015.

Validation Message: A blue arrow points from the 'Add a Grade' section down to a modal dialog box. The dialog has an information icon and the text 'A grade is already Assigned to that subject'. It has an 'OK' button at the bottom. A green oval highlights this dialog box.

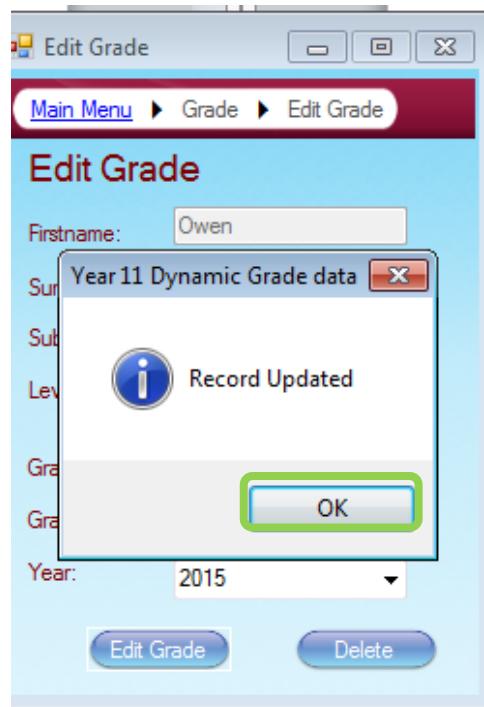
Reference 58

Evidence which shows what happens when editing a grade.

The image displays two screenshots of a Windows application window titled "Edit Grade". Both screenshots show the same form fields: Firstname (Owen), Surname (Joel), Subject (Drama), Level (KS4), Grade (dropdown menu), Grade Type (Achieved), and Year (2015). In the first screenshot, the Grade dropdown contains "D". In the second screenshot, the Grade dropdown contains "G". The Grade dropdown in both screenshots is highlighted with a green border. A blue arrow points from the top window down to the bottom window, indicating the transition or result of the edit.

Field	Value - Top Window	Value - Bottom Window
Firstname	Owen	Owen
Surname	Joel	Joel
Subject	Drama	Drama
Level	KS4	KS4
Grade	D	G
Grade Type	Achieved	Achieved
Year	2015	2015

Reference 58 (continued)



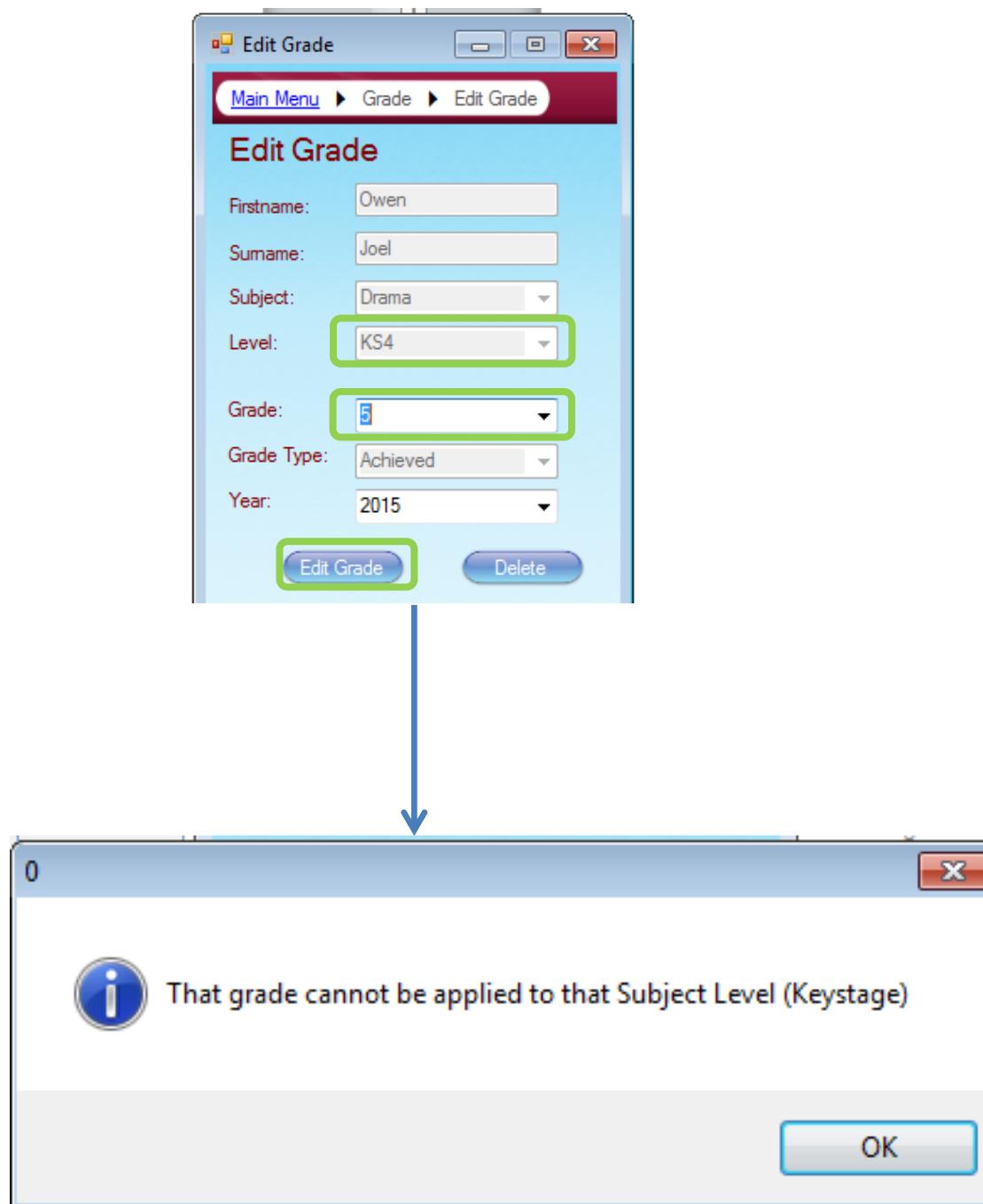
Evidence of record updated in SQL server database.

The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. On the left is the Object Explorer showing database objects like tables, views, and stored procedures. In the center, a query results grid displays data from a 'Grade' table. The grid has columns: GradeID, StudentID, SubjectID, LevelID, GradeTypeID, Grade, and DatasetID. The data shows several rows of student grades. To the right of the results grid is a Properties pane showing connection details for a session named 'STUDENTDEV'. The session details include connection time, rows returned, and session ID. The status bar at the bottom indicates the query was executed successfully.

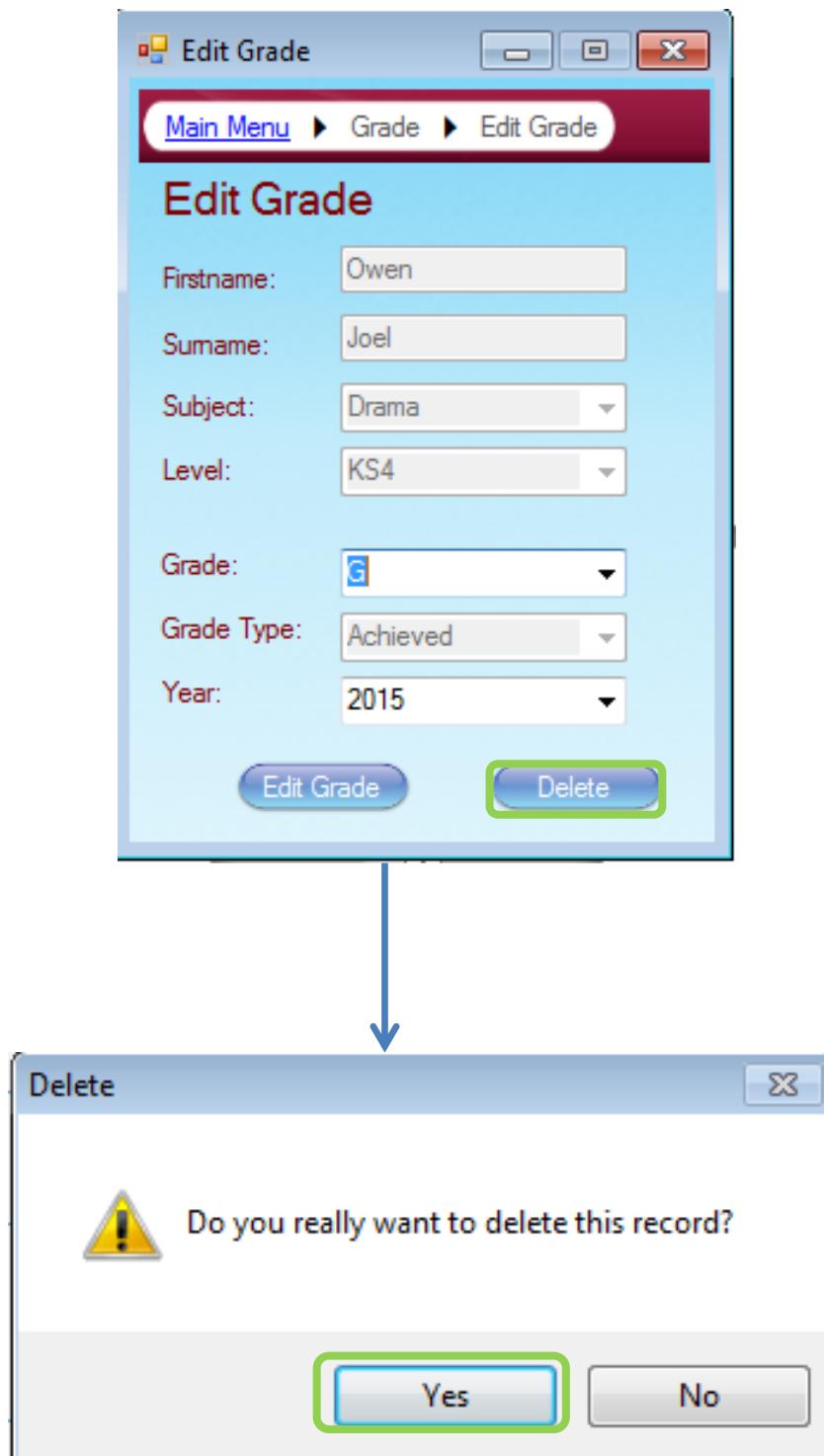
GradeID	StudentID	SubjectID	LevelID	GradeTypeID	Grade	DatasetID
6768	552	13	1	1	4	1
6769	553	13	1	1	4	1
6770	554	13	1	2	3	3
6771	555	13	1	2	3	3
6772	99	1	2	1	A*	1
6769	553	13	1	1	4	1
6768	552	13	1	1	4	1
6767	551	13	1	1	4	1
6773	224	3	2	1	G	3

Reference 59

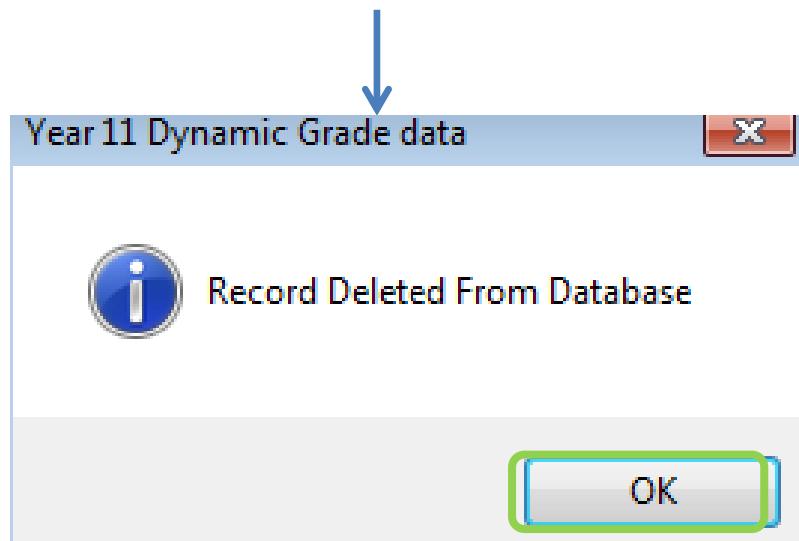
Evidence which shows validation of the system to prevent data inconsistency.



Reference 60



Reference 60 (Continued)



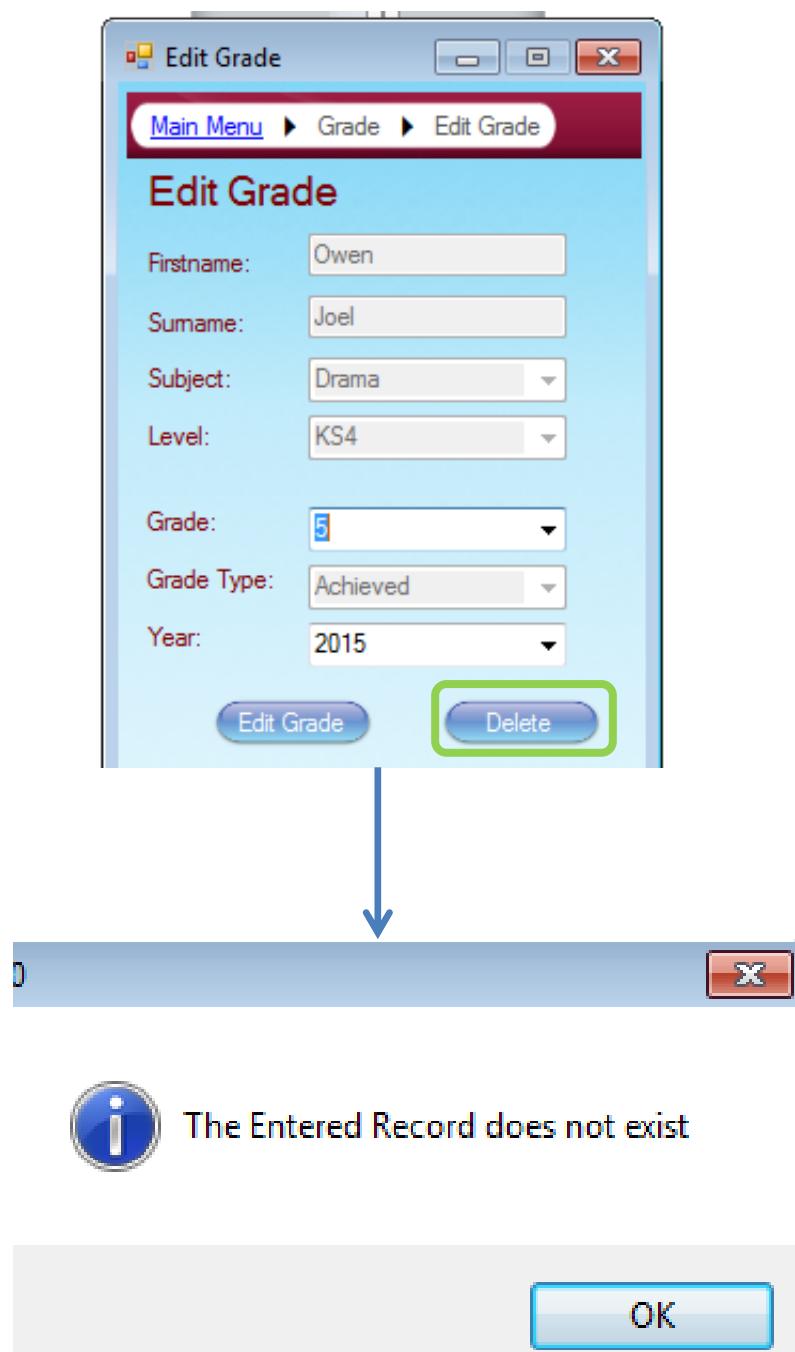
Evidence to show the grade record has been deleted from the SQL server database. It's grade ID was 6768 but that record no longer exists.

GradeID	StudentID	SubjectID	LevelID	GradeTypeID	Grade	DatasetID
6757	541	13	1	1	4	1
6758	542	13	1	1	5	1
6759	543	13	1	1	4	1
6760	544	13	1	1	5	1
6761	545	13	1	1	4	1
6762	546	13	1	1	5	1
6763	547	13	1	1	3	1
6764	548	13	1	1	4	1
6765	549	13	1	1	5	1
6767	551	13	1	1	4	1
6768	552	13	1	1	4	1
6769	553	13	1	1	4	1
6770	554	13	1	2	3	3
6771	555	13	1	2	3	3
6772	99	1	2	1	A*	1
6769	553	13	1	1	4	1
6768	552	13	1	1	4	1
6767	551	13	1	1	4	1
6766	550	13	1	1	4	1

Query executed successfully.

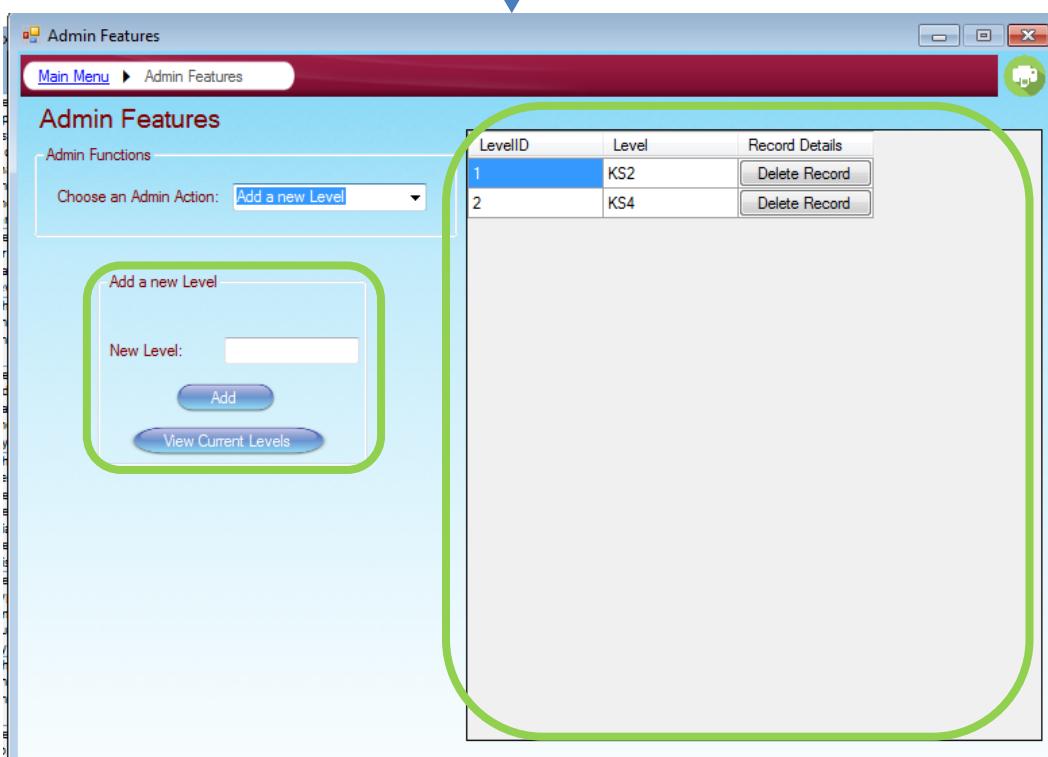
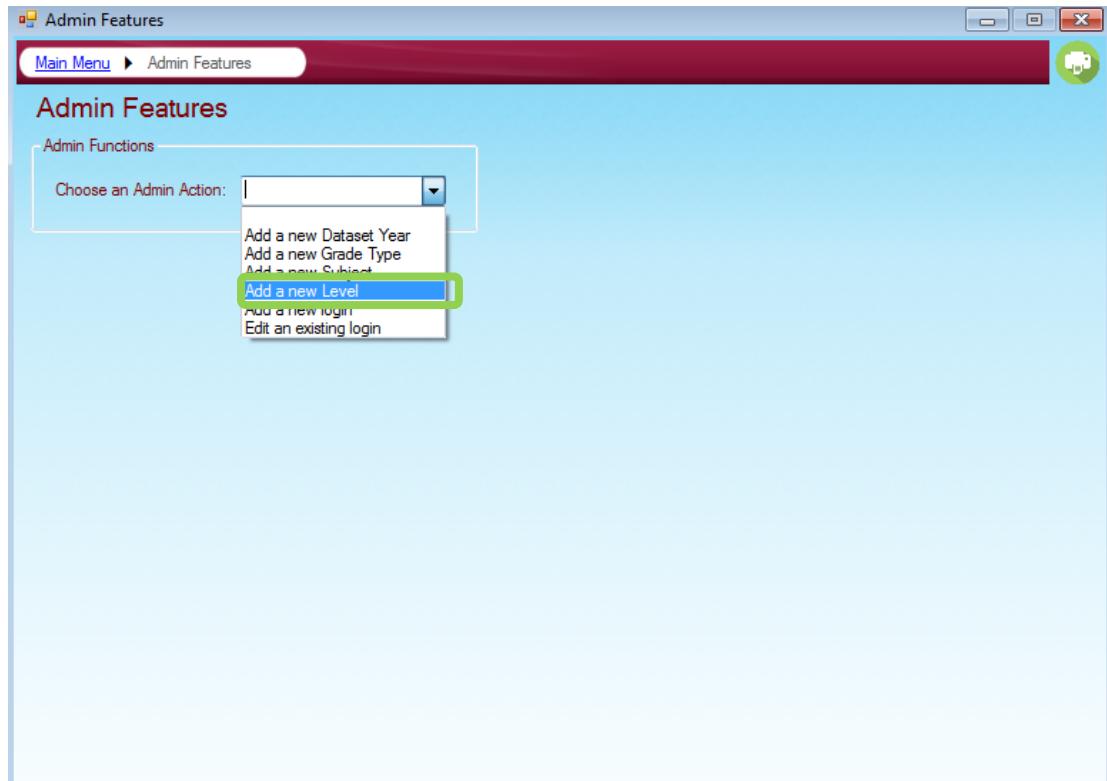
Reference 61

Evidence which shows validation of the system.



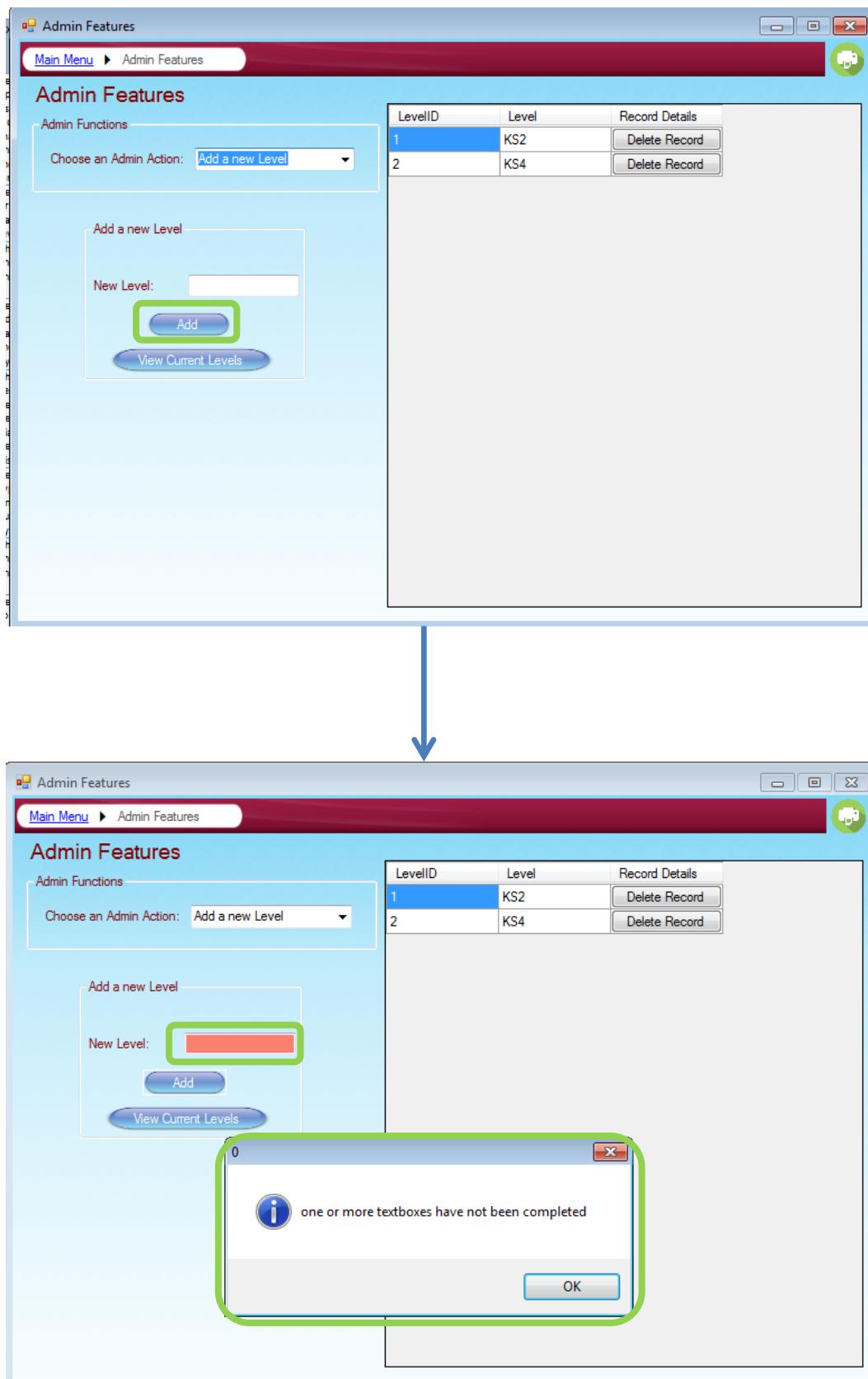
Reference 62

Evidence to demonstrate the functionality of my admin features combo selection.



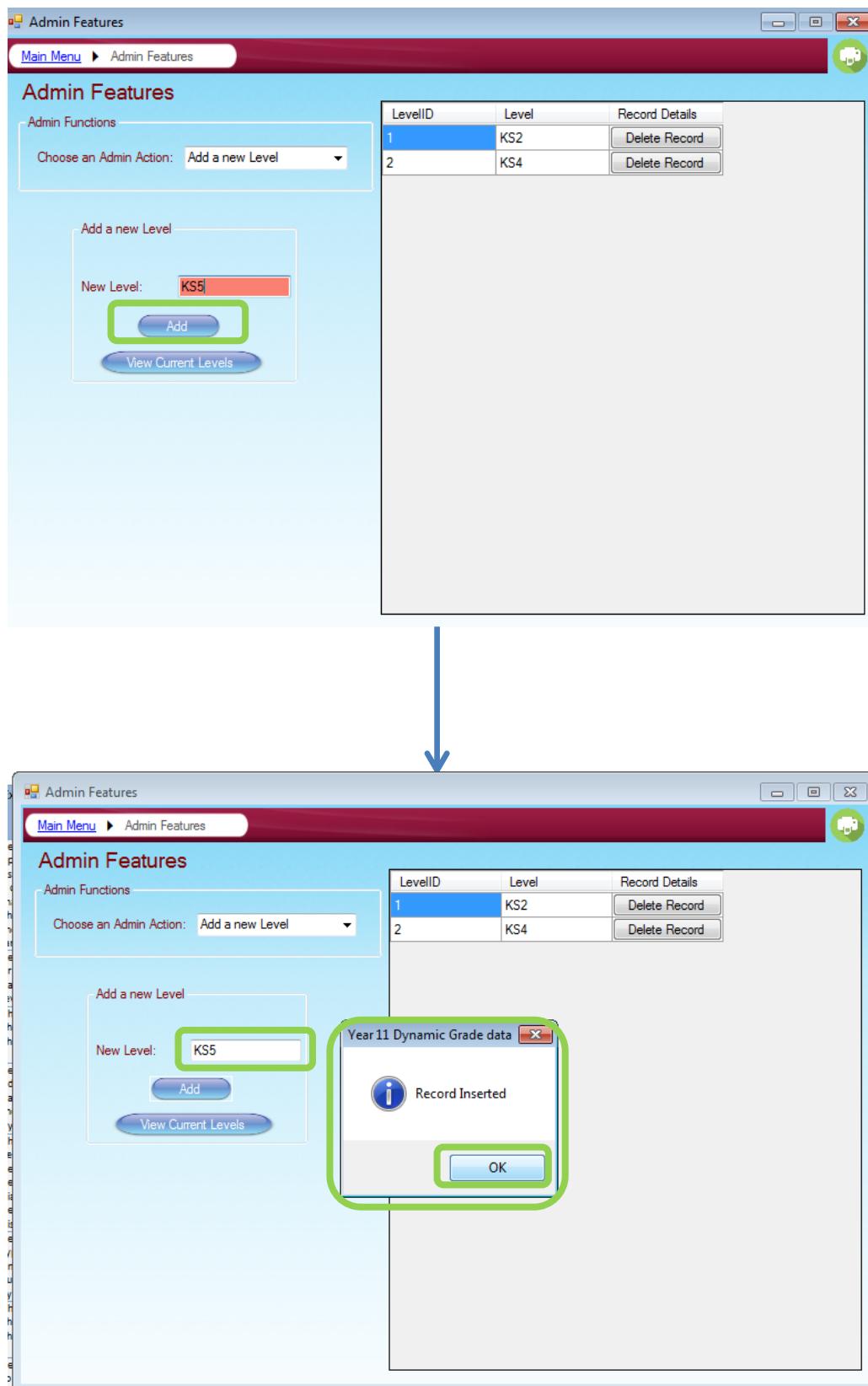
Reference 63

Evidence which demonstrates the validation of my system.



Reference 64

Test evidence which demonstrates the process of using admin insert features.



The screenshot shows a Windows application window titled "Admin Features". On the left, there's a sidebar with "Admin Functions" and a dropdown menu set to "Add a new Level". Below it is a form titled "Add a new Level" with a text input field containing "New Level: KS5" and two buttons: "Add" and "View Current Levels". On the right, there's a table titled "Record Details" with three rows:

LevelID	Level	Record Details
1	KS2	Delete Record
2	KS4	Delete Record
3	KS5	Delete Record

A blue arrow points down from the top of the page towards the application window.

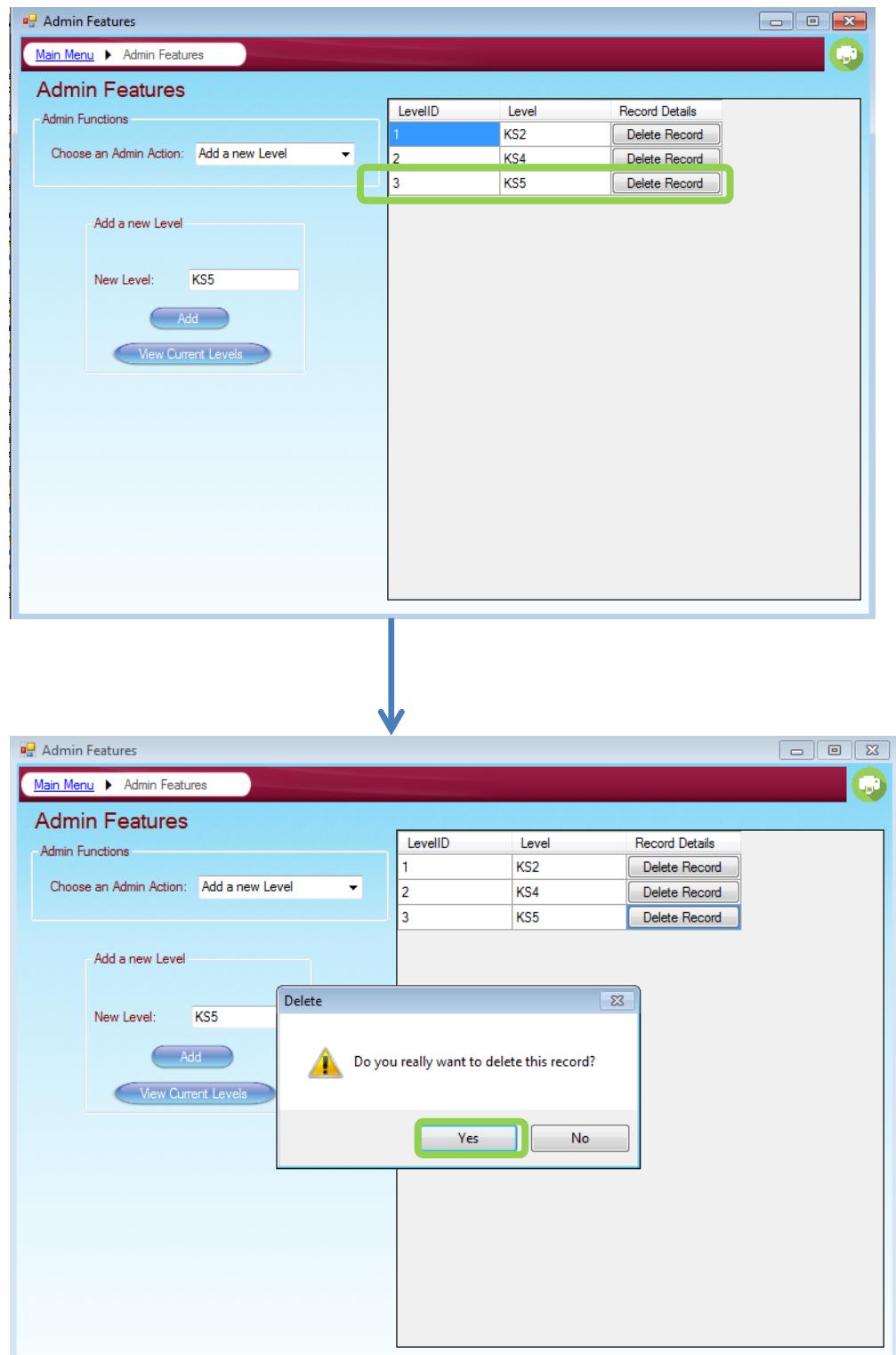
Evidence of the record being inserted into the SQL server database successfully.

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the database "dn.singleton" is selected. In the center pane, a query results grid displays the following data:

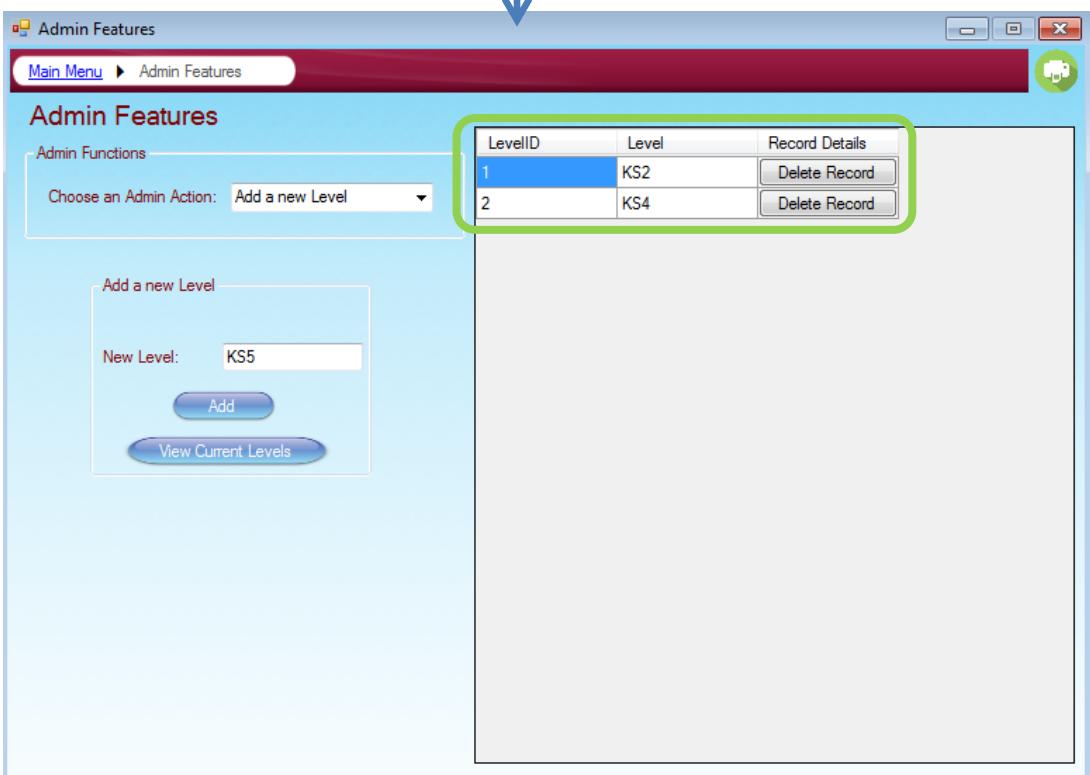
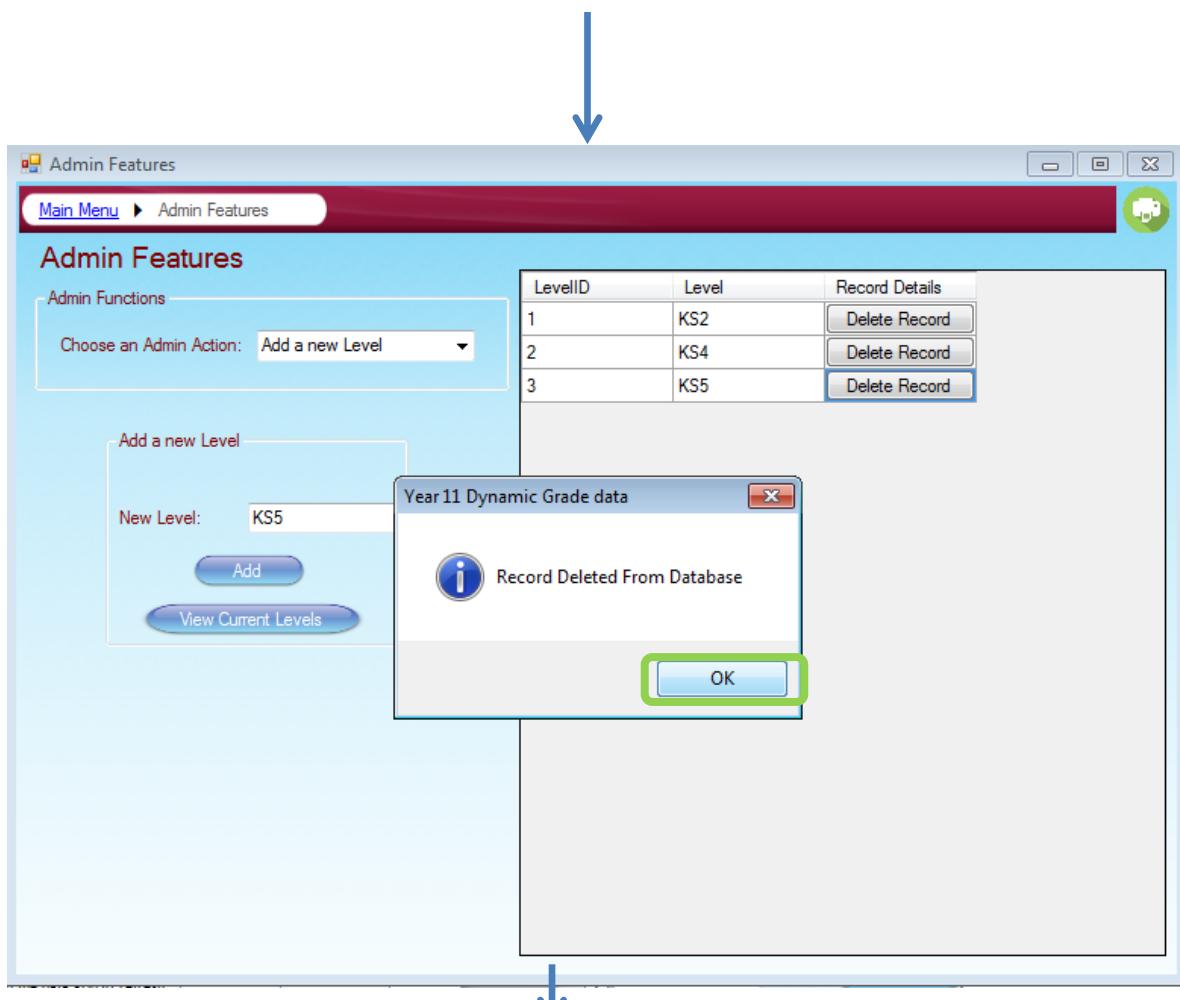
LevelID	Level
1	KS2
2	KS4
3	KS5

The results grid has a green outline around its border. The status bar at the bottom of the screen shows the message "Query executed successfully." and the connection details: STUDENTDEV (10.50 RTM) | dn.singleton (116) | dn.singleton | 00:00:00 | 3 rows.

Reference 65

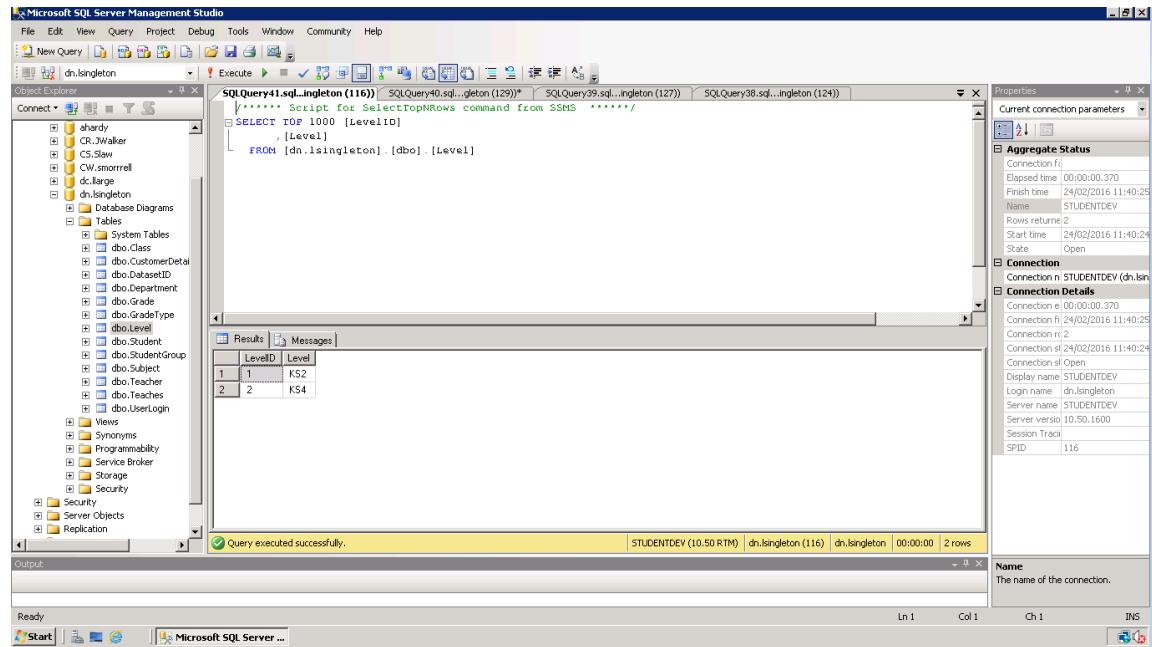


Reference 65 (continued)



Reference 65 (continued)

Evidence which shows that the record has been deleted from the SQL server database successfully.



The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows a database structure with several schemas and tables. The central pane displays a query results grid and the T-SQL code used to retrieve data. The Properties pane on the right provides connection details. The status bar at the bottom indicates the query was executed successfully.

Object Explorer:

- shady
- CR_Walker
- CS_Slava
- CW_smorrell
- dc_large
- dn.singleton
- Database Diagrams
- Tables
 - System Tables
 - dbo_Class
 - dbo_CustomerDetail
 - dbo_DatasetID
 - dbo_Department
 - dbo_Grade
 - dbo_GradeType
 - dbo_Level
 - dbo_Student
 - dbo_StudentGroup
 - dbo_Subject
 - dbo_Teacher
 - dbo_Values
 - dbo_UserLogin
- Views
- Synonyms
- Programmability
- Service Broker
- Storage
- Security
- Server Objects
- Replication

SQL Query Results:

```
***** Script for SelectTopNRows command from SSMS *****
SELECT TOP 1000 [LevelID]
      ,[Level]
  FROM [dn.singleton].[dbo].[Level]
```

LevelID	Level
1	KS2
2	KS4

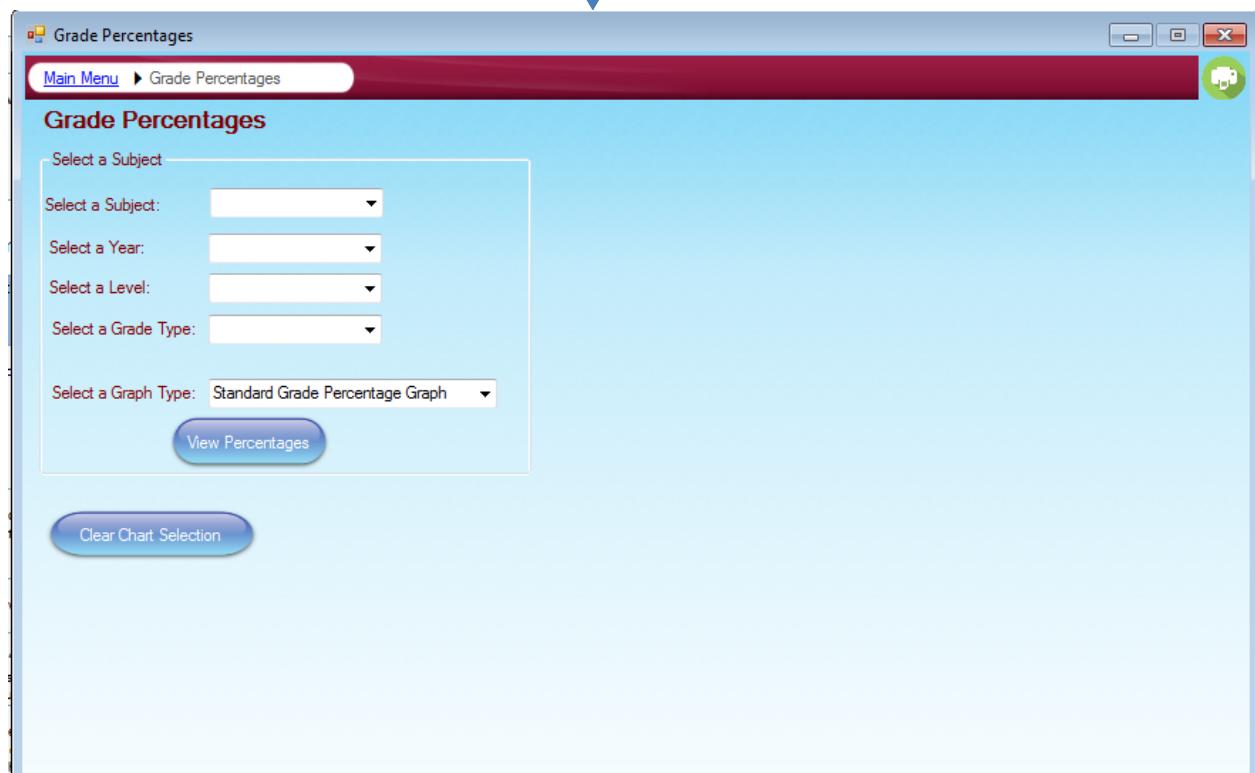
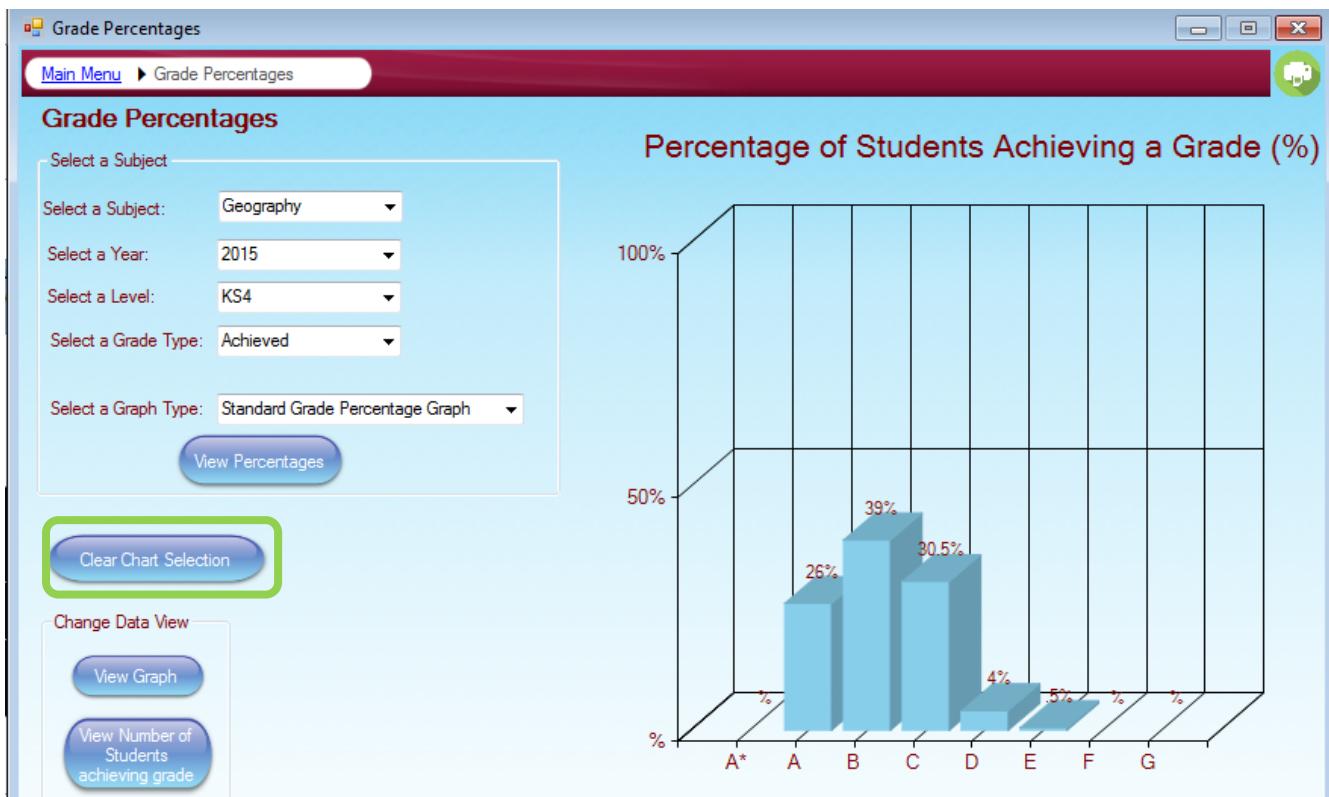
Status Bar: Query executed successfully.

Properties pane:

- Aggregate Status
 - Connection #1
 - Elapsed time: 00:00:00.370
 - Finish time: 24/02/2016 11:40:25
 - Name: STUDENTDEV
 - Rows returned: 2
 - Start time: 24/02/2016 11:40:24
 - State: Open
- Connection
 - Connection n: STUDENTDEV (dn.singleton)
 - Connection o: 00:00:00.370
 - Connection r: 24/02/2016 11:40:25
 - Connection s: 24/02/2016 11:40:24
 - Connection t: Open
 - Display name: STUDENTDEV
 - Login name: dn.singleton
 - Server name: STUDENTDEV
 - Server version: 10.50.1600
 - Session Tracel
 - SPID: 116

Reference 66

Evidence which shows the functionality of the clear chart selection button.



Reference 67

Evidence which shows evidence of the view graph's button functionality.

Grade Percentages

Main Menu ▶ Grade Percentages

Grade Percentages

Select a Subject

Select a Subject: English Literature

Select a Year: 2015

Select a Level: KS4

Select a Grade Type: Achieved

Select a Graph Type: Standard Grade Percentage Graph

View Percentages

Clear Chart Selection

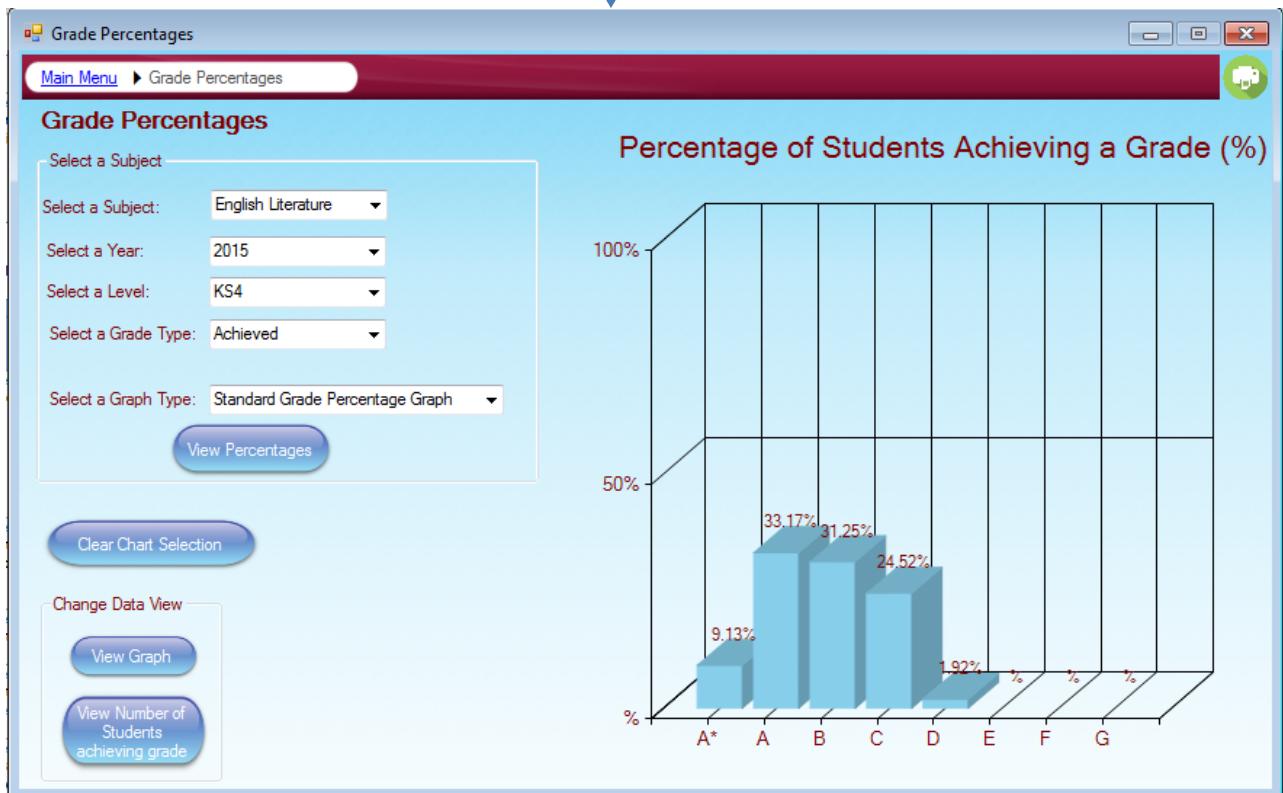
Change Data View

View Graph

View Number of Students achieving grade

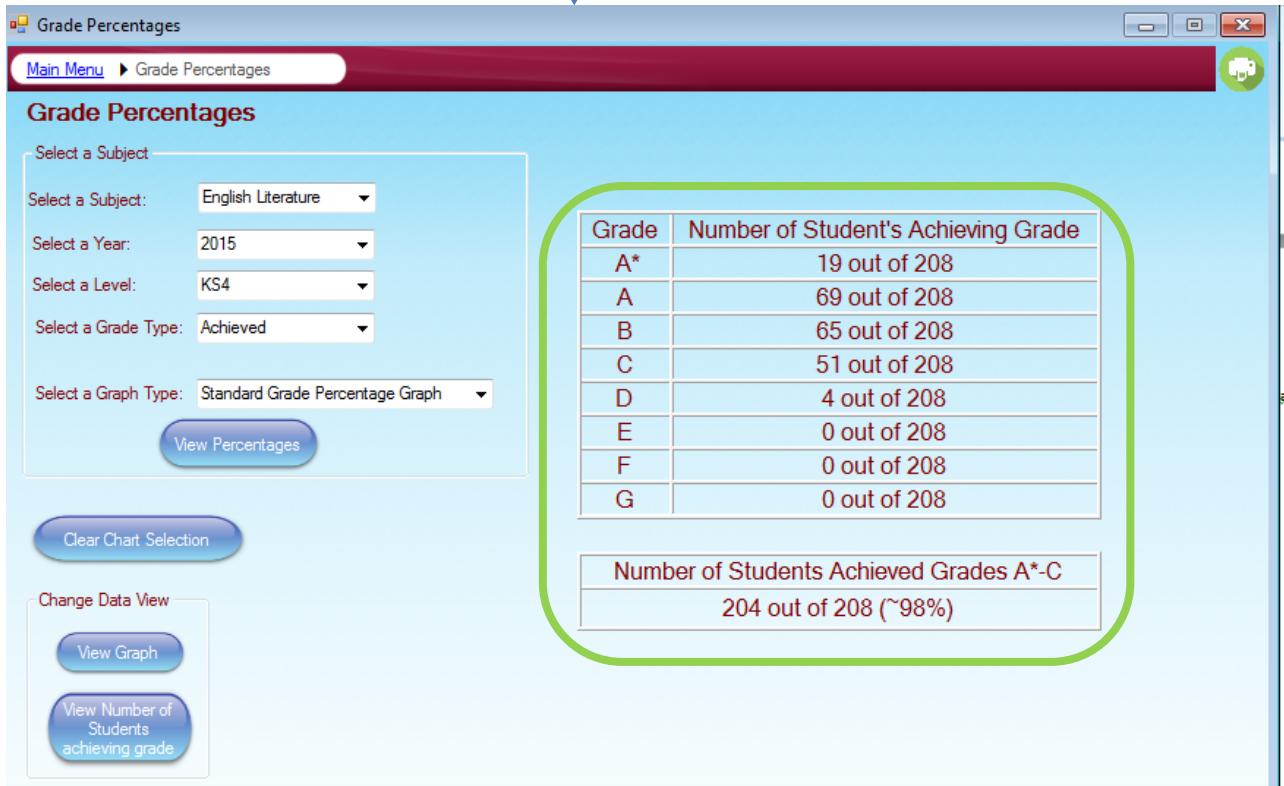
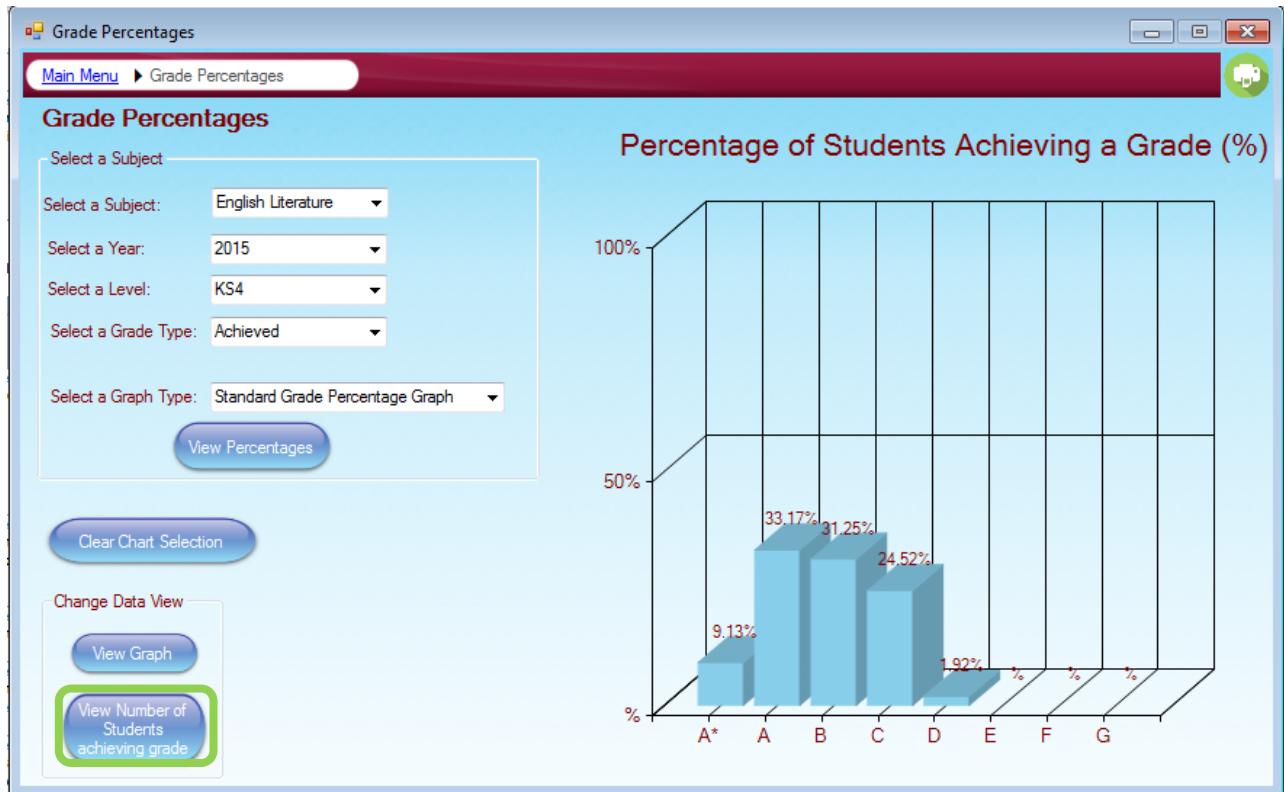
Grade	Number of Student's Achieving Grade
A*	19 out of 208
A	69 out of 208
B	65 out of 208
C	51 out of 208
D	4 out of 208
E	0 out of 208
F	0 out of 208
G	0 out of 208

Number of Students Achieved Grades A*-C
204 out of 208 (~98%)



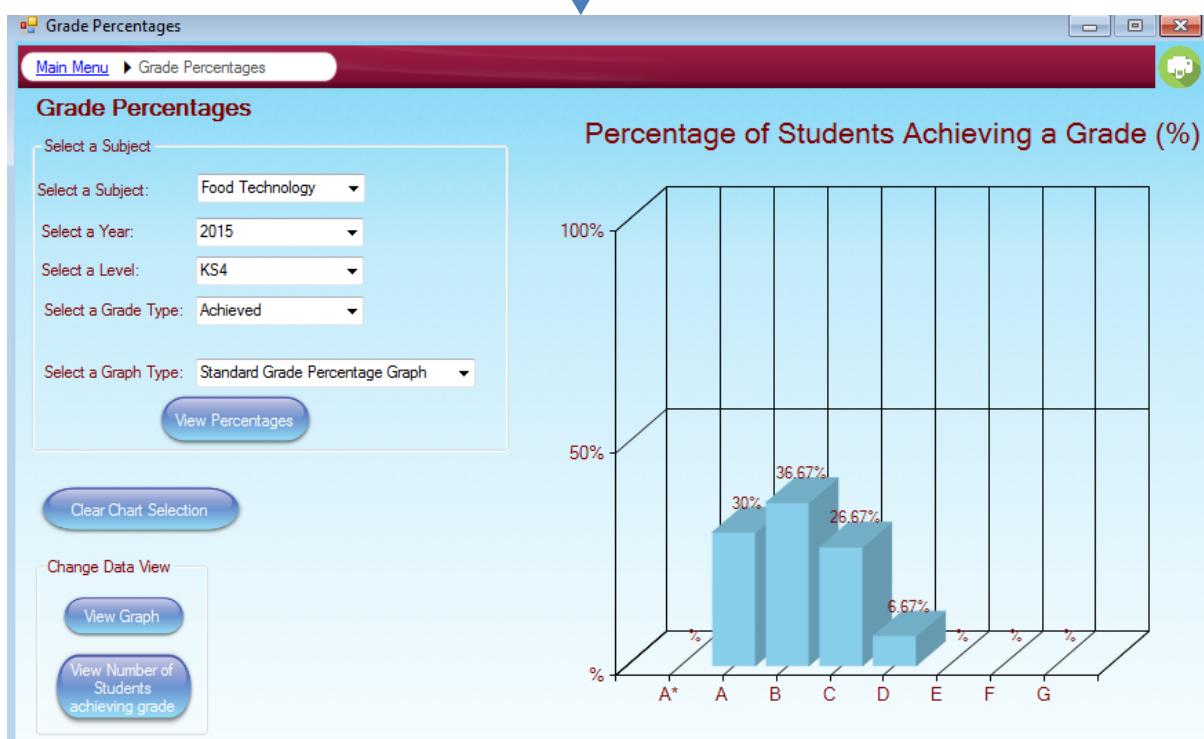
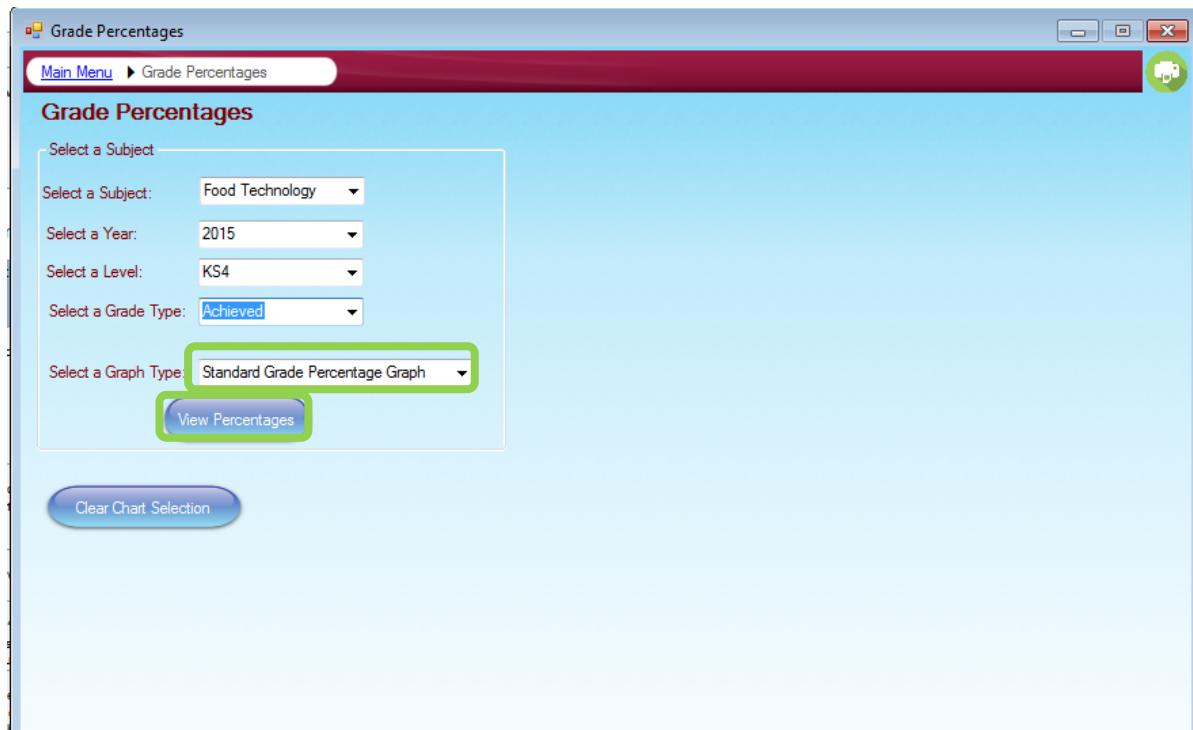
Reference 68

Evidence which shows evidence of the view raw data button functionality.



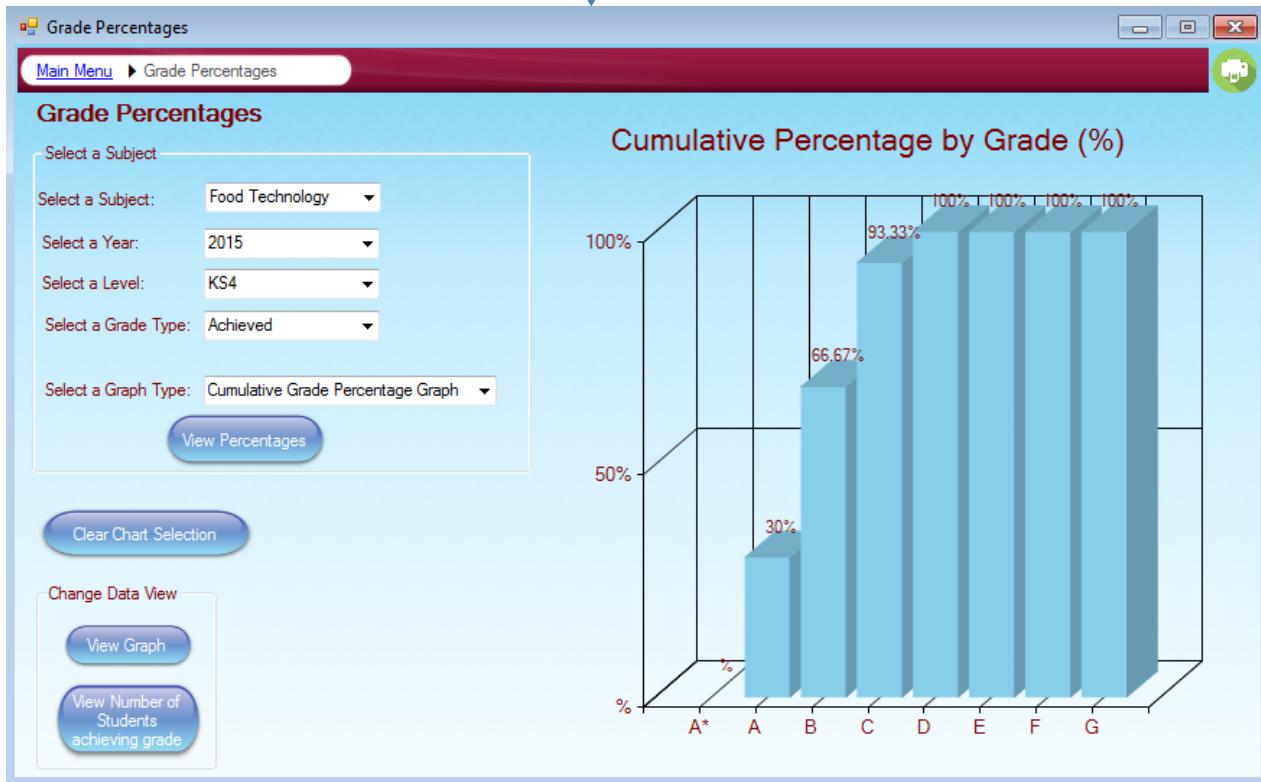
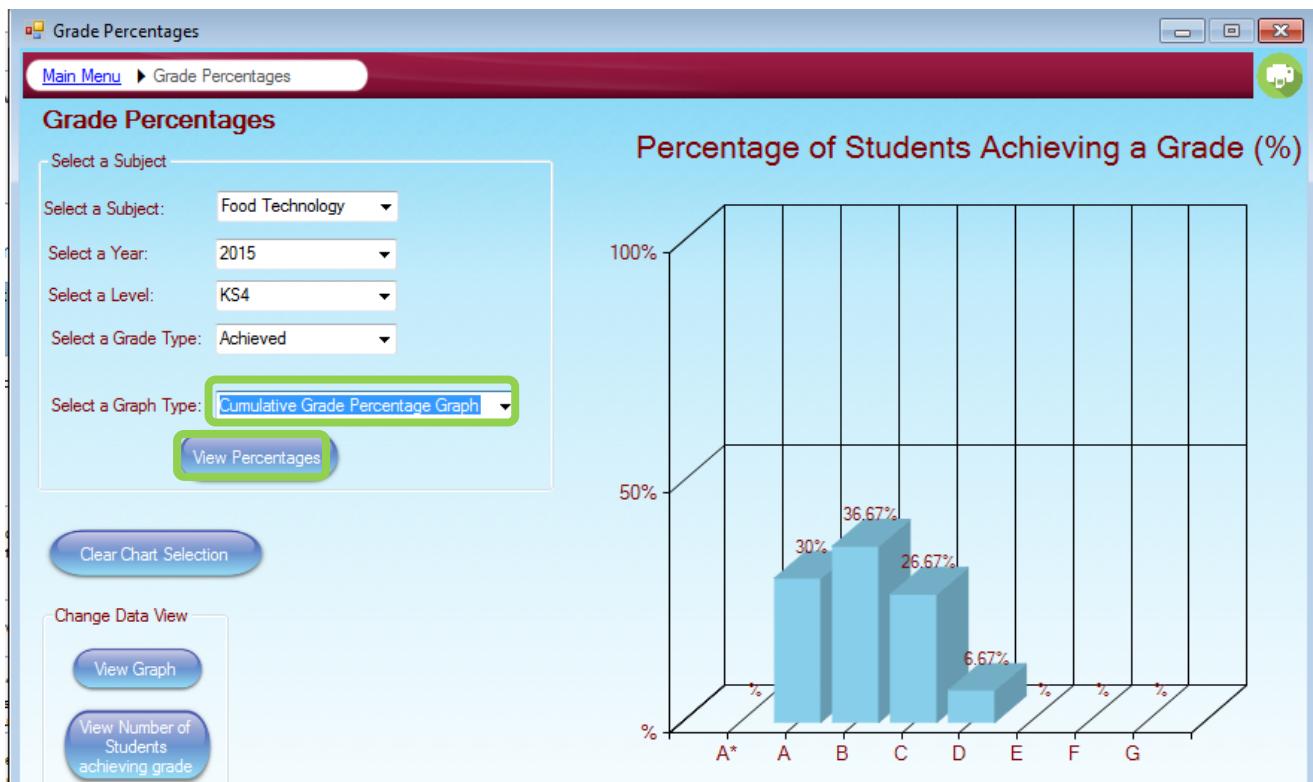
Reference 69

Evidence which shows one of the two different graph types a user can use.



Reference 70

Evidence which shows one of the two different graph types a user can use.



Reference 71

Note that the grade type field is enabled when “KS4” is selected because predicted grades can be viewed based on “KS4” grades.

The screenshot shows a Windows application window titled "Grade Percentages". The window has a blue header bar with the title and a red navigation bar below it containing "Main Menu" and "Grade Percentages". On the right side of the header bar is a green circular icon with a white hand symbol. The main content area is titled "Grade Percentages" and contains the following form fields:

- Select a Subject: A dropdown menu.
- Select a Year: A dropdown menu.
- Select a Level: A dropdown menu with "KS4" selected, highlighted by a green rounded rectangle.
- Select a Grade Type: A dropdown menu.
- Select a Graph Type: A dropdown menu set to "Standard Grade Percentage Graph".
- A blue "View Percentages" button.
- A blue "Clear Chart Selection" button at the bottom left.

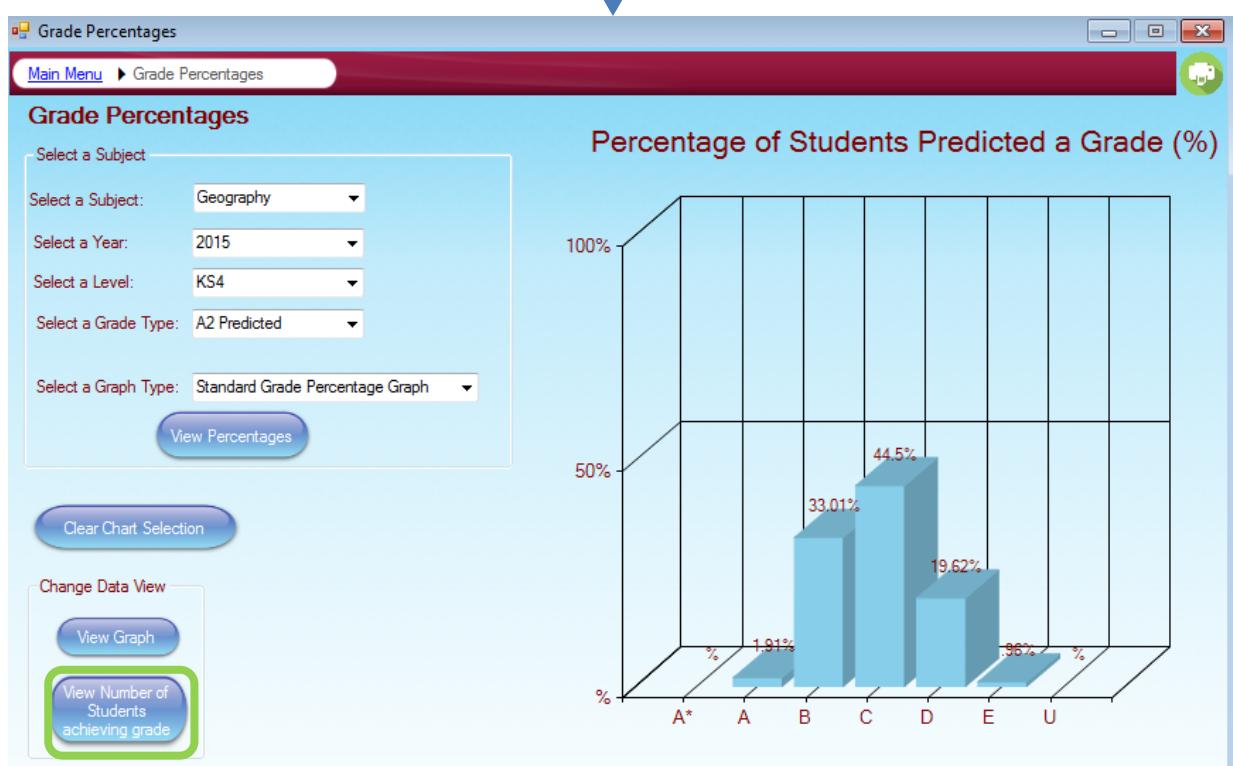
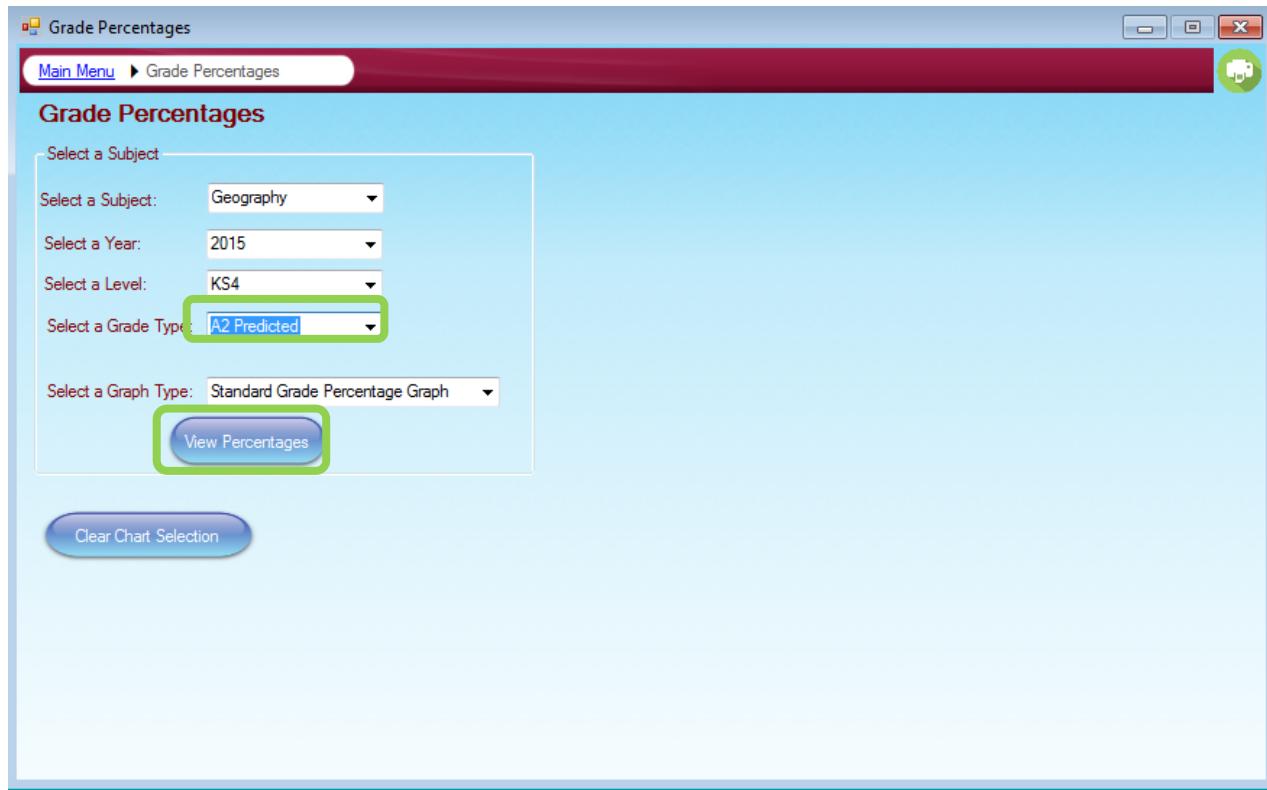
Reference 72

Note that when a user selects “KS2” as the selected level that the grade type field is not enabled and its default text is set to “Achieved”. This is because a user can not view predicted grades based on KS2 grades.

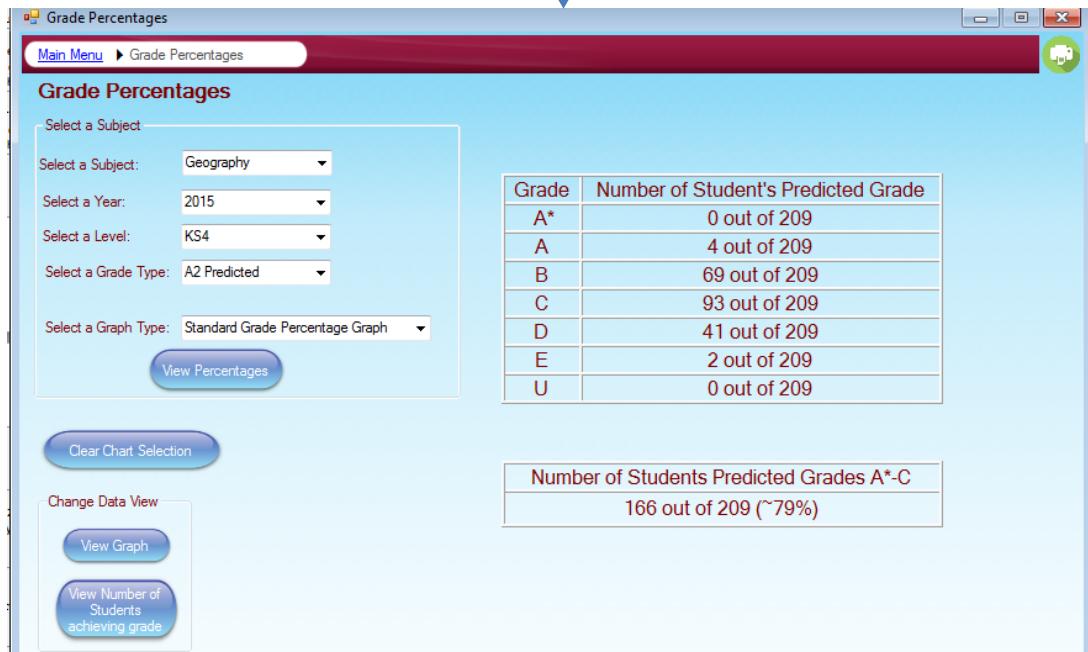
The screenshot shows a Windows application window titled "Grade Percentages". The window has a red header bar with "Main Menu" and "Grade Percentages" buttons, and a green camera icon. Below the header is a title "Grade Percentages". A "Select a Subject" section contains four dropdown menus: "Select a Subject", "Select a Year", "Select a Level" (which is set to "KS2" and highlighted with a green border), and "Select a Grade Type" (which is set to "Achieved"). Below these is a "Select a Graph Type" dropdown set to "Standard Grade Percentage Graph" and a "View Percentages" button. At the bottom left is a "Clear Chart Selection" button.

Reference 73

This evidence shows the functionality of generating Predicted grades for an entire subject.

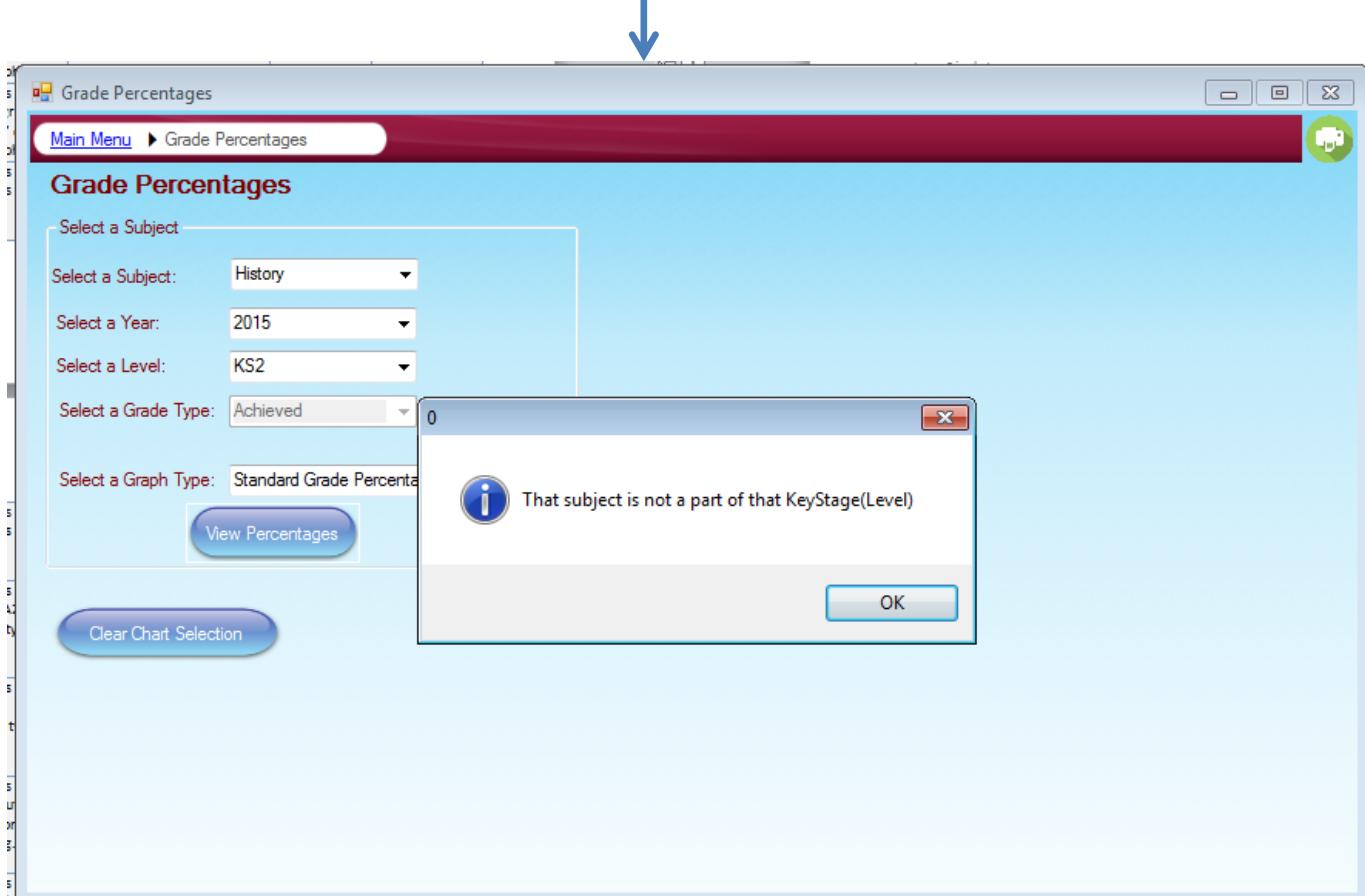
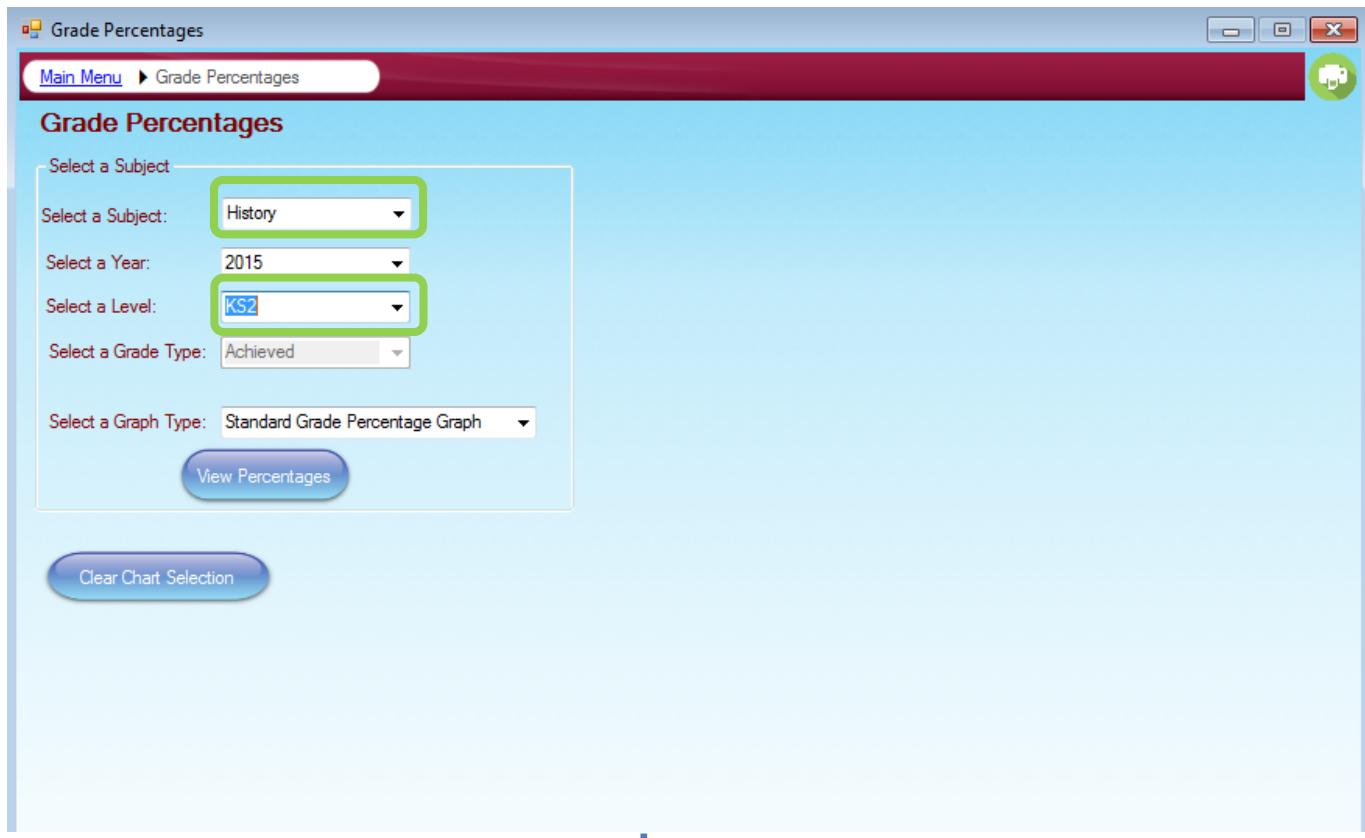


Reference 73 (continued)



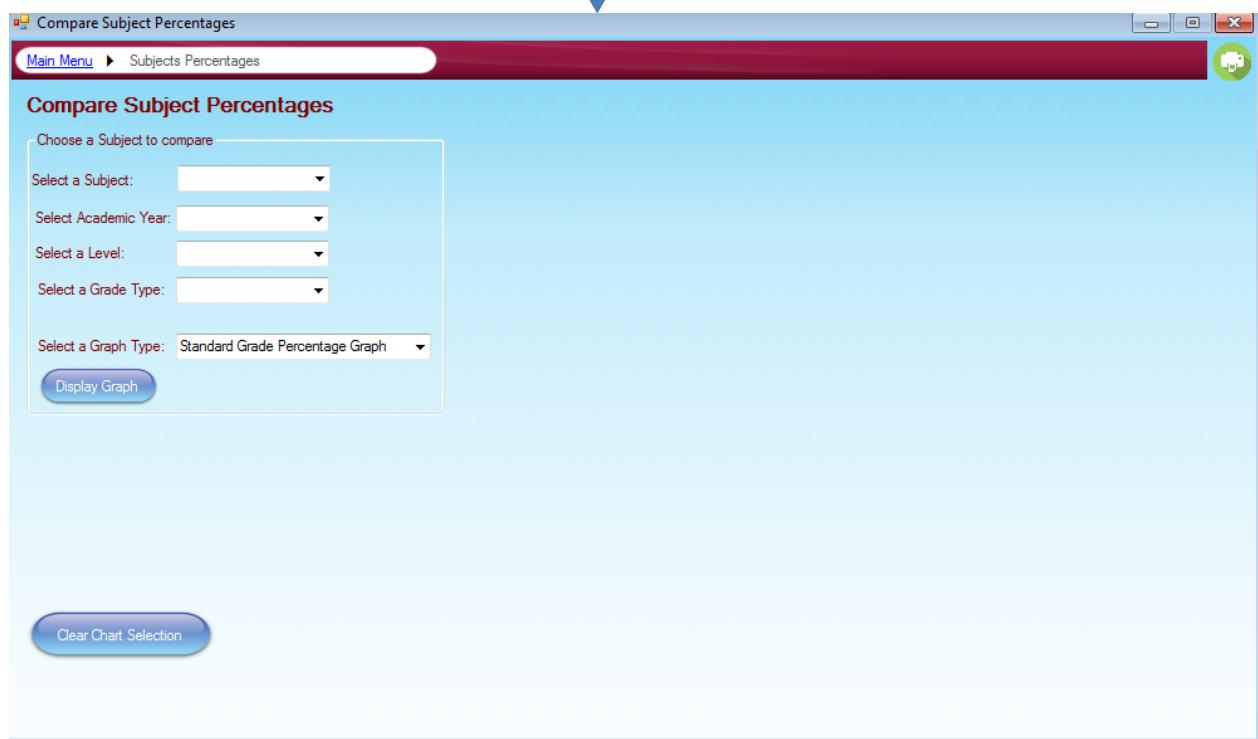
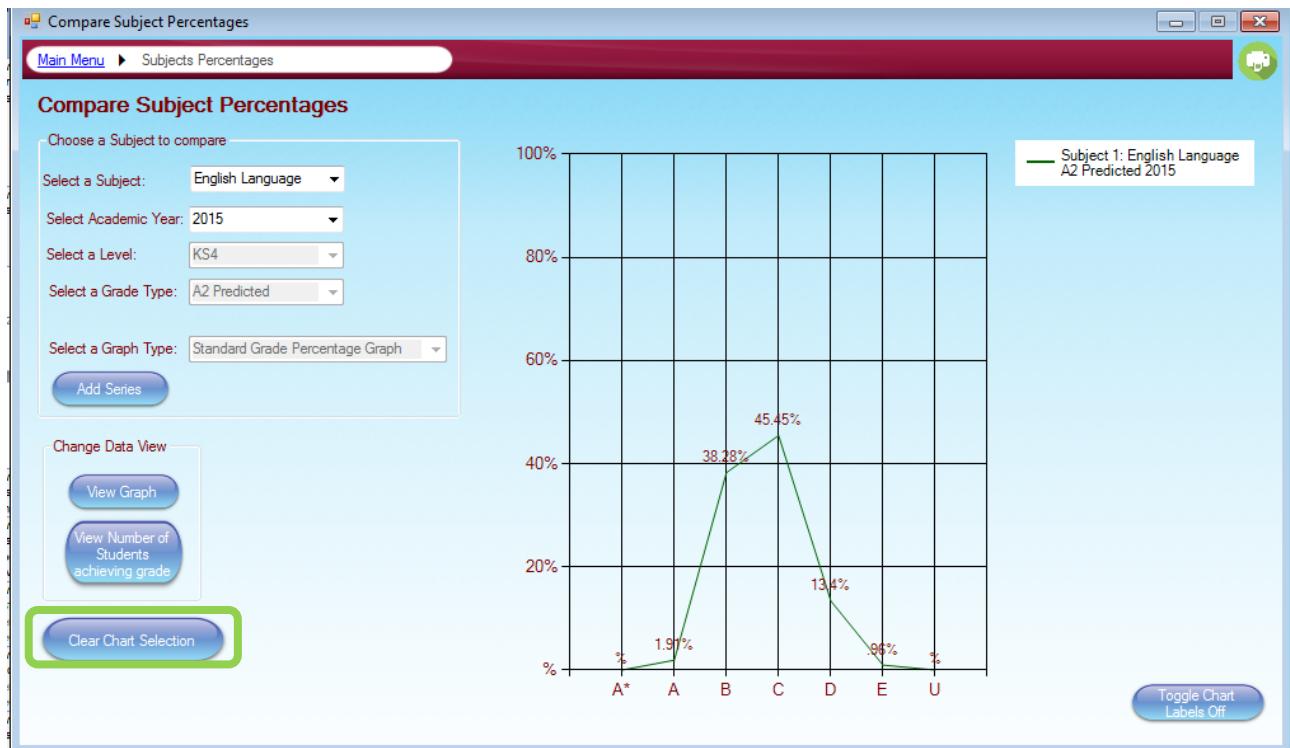
Reference 74

Evidence which shows validation within my system.



Reference 75

Evidence which shows the functionality of the “clear chart selection” button.



Reference 76

Evidence which demonstrates the functionality of the view graph button.

Compare Subject Percentages

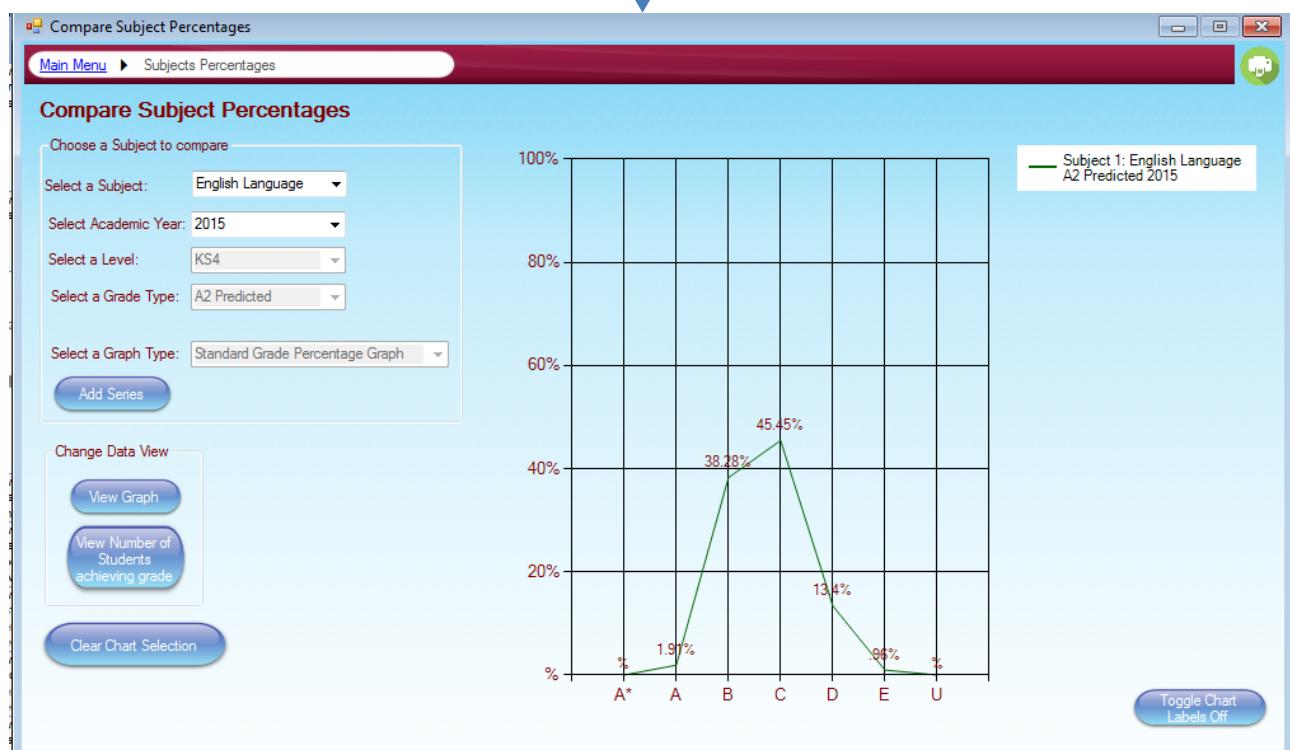
Subject 1: English Language A2 Predicts 2015

Grade	Number of Student's Predicted Grade
A*	0 out of 209
A	4 out of 209
B	80 out of 209
C	95 out of 209
D	28 out of 209
E	2 out of 209
U	0 out of 209
A*-C	179 out of 209 (~85%)

Change Data View

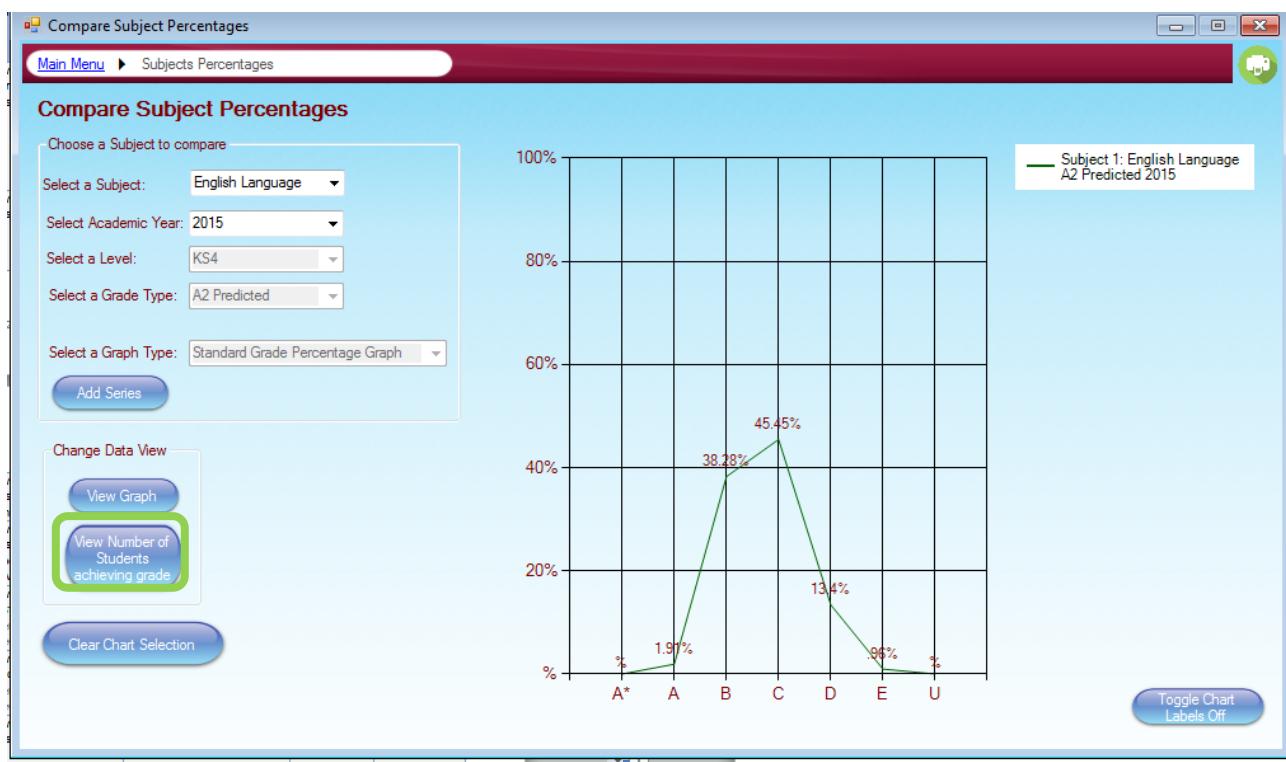
- [View Graph](#)
- [View Number of Students achieving grade](#)

[Clear Chart Selection](#)



Reference 77

Evidence which demonstrates the functionality of the view raw statistics button.



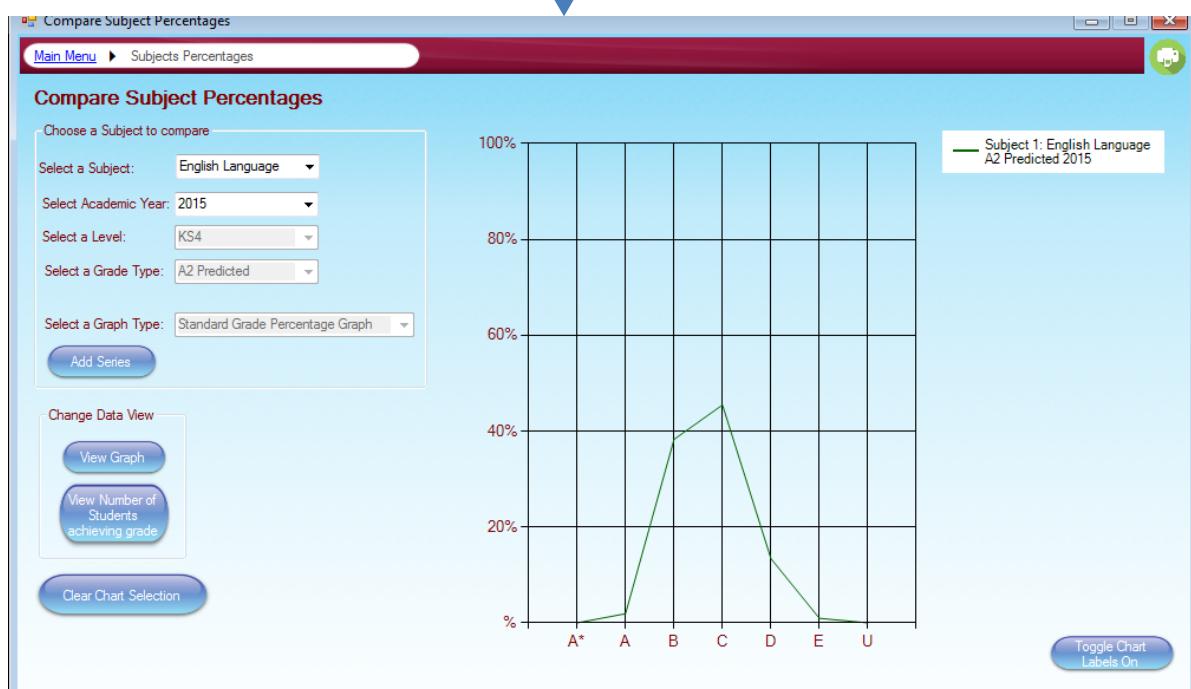
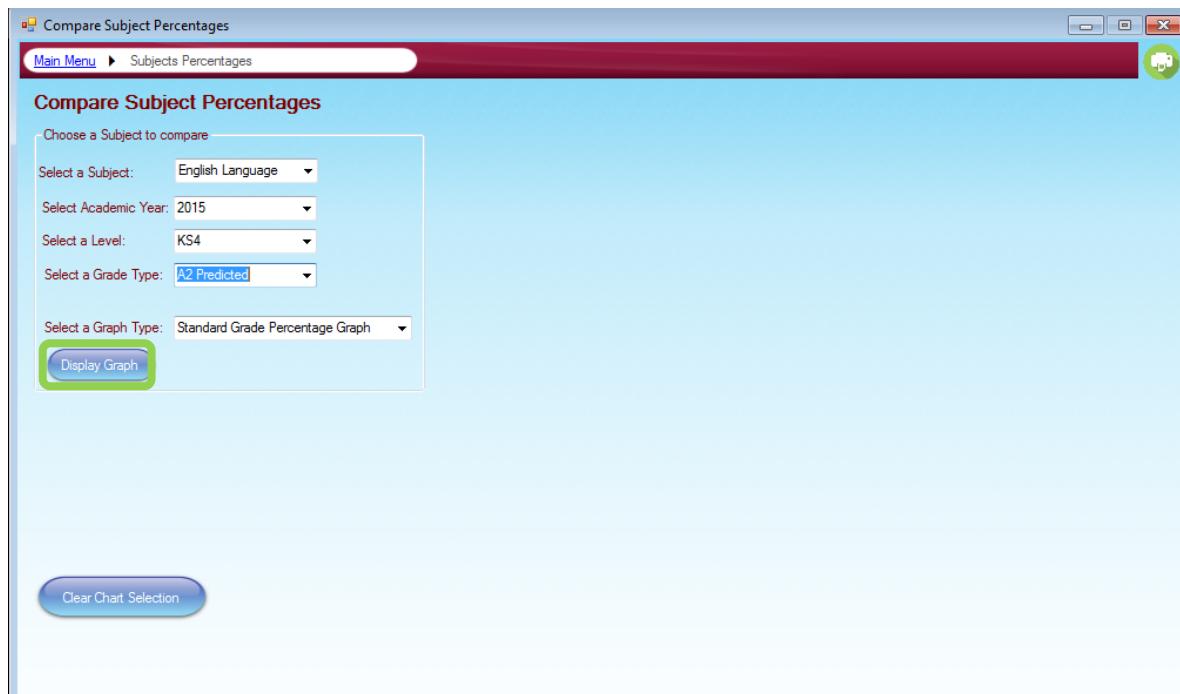
The screenshot shows the same application window after clicking the 'View Number of Students achieving grade' button. The title bar now includes 'Subject 1: English Language A2 Predicted 2015'. The main area displays a table titled 'Subject 1: English Language A2 Predicts 2015' with the following data:

Grade	Number of Student's Predicted Grade
A*	0 out of 209
A	4 out of 209
B	80 out of 209
C	95 out of 209
D	28 out of 209
E	2 out of 209
U	0 out of 209
A*-C	179 out of 209 (~85%)

The sidebar on the left remains the same, with 'View Graph' and 'View Number of Students achieving grade' buttons, and a 'Clear Chart Selection' button. A blue arrow points upwards from the previous window to this one.

Reference 78

Evidence which shows a linear graph type in use.



Reference 79

Evidence which demonstrates the functionality of the Cumulative graph series.

Compare Subject Percentages

Main Menu > Subjects Percentages

Compare Subject Percentages

Choose a Subject to compare

Select a Subject: English Language

Select Academic Year: 2015

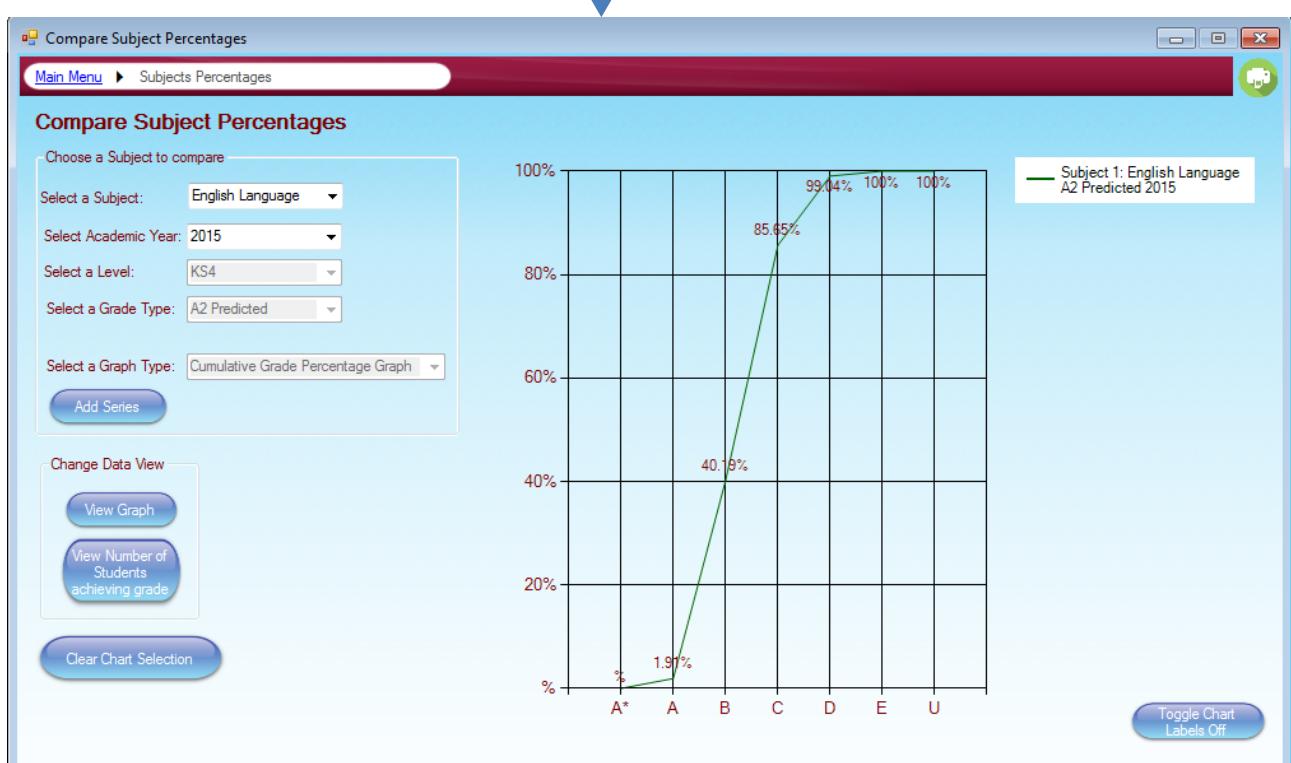
Select a Level: KS4

Select a Grade Type: A2 Predicted

Select a Graph Type: Cumulative Grade Percentage Graph

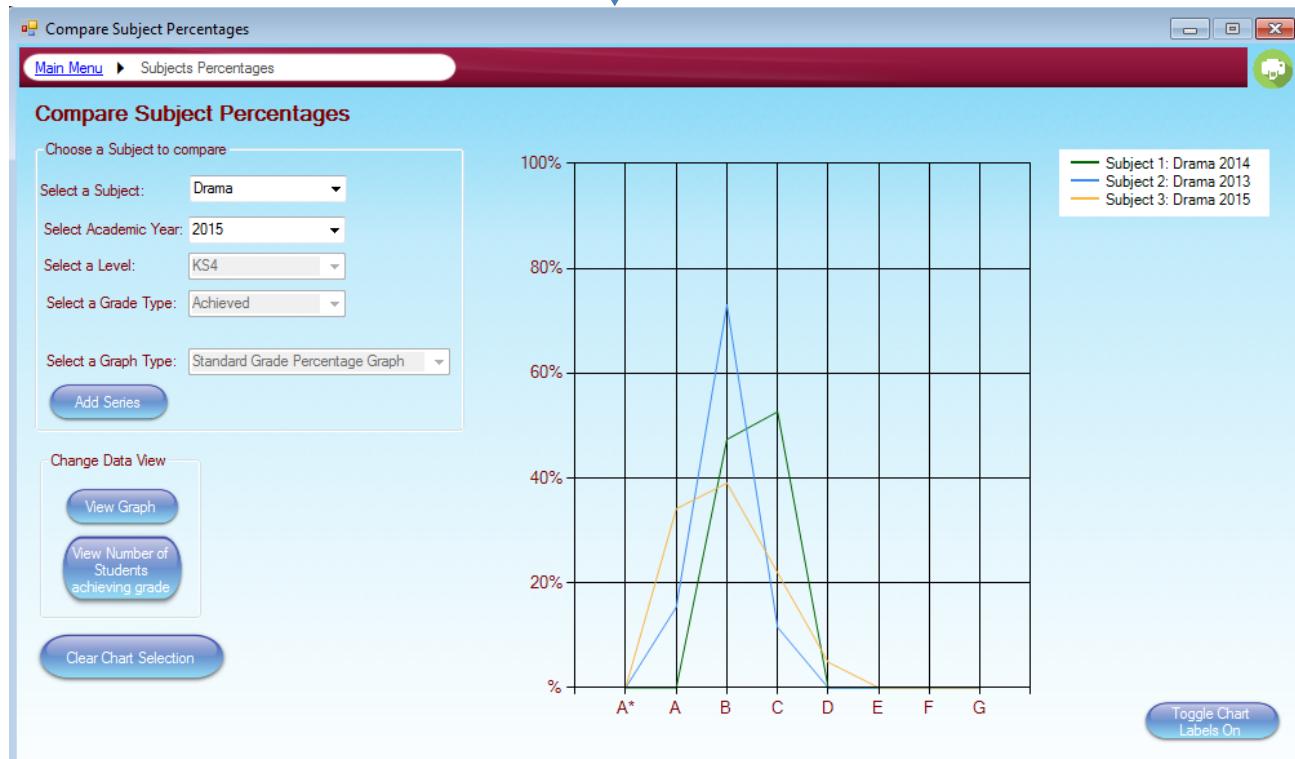
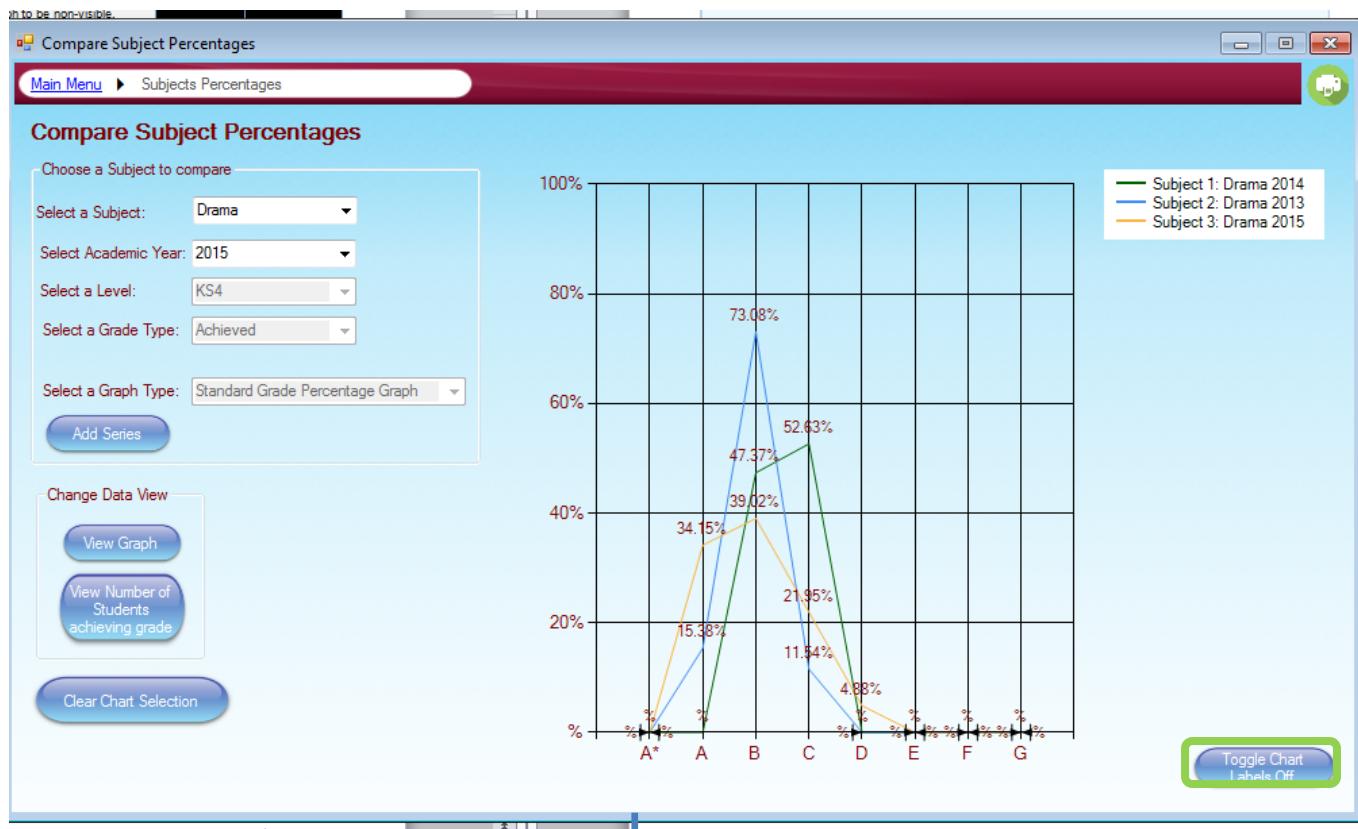
Display Graph

Clear Chart Selection



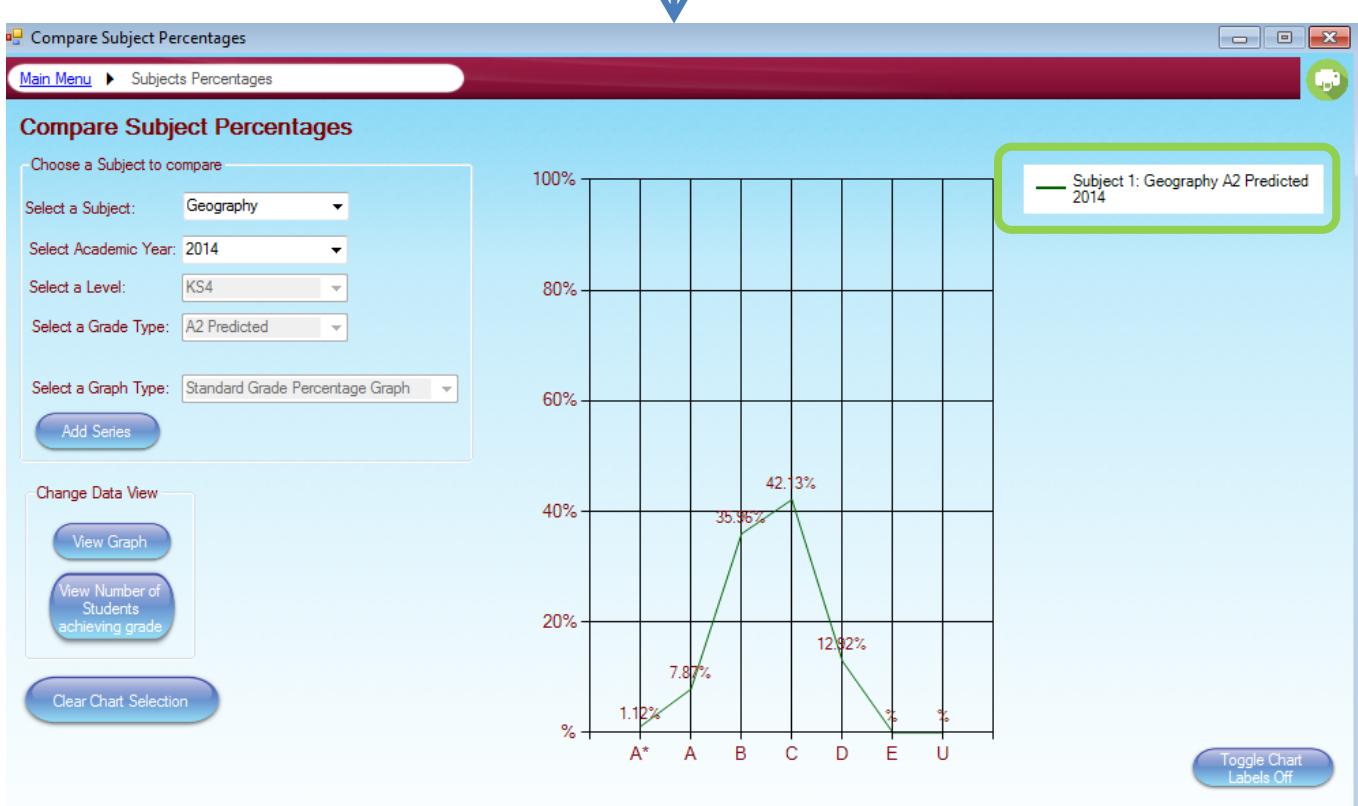
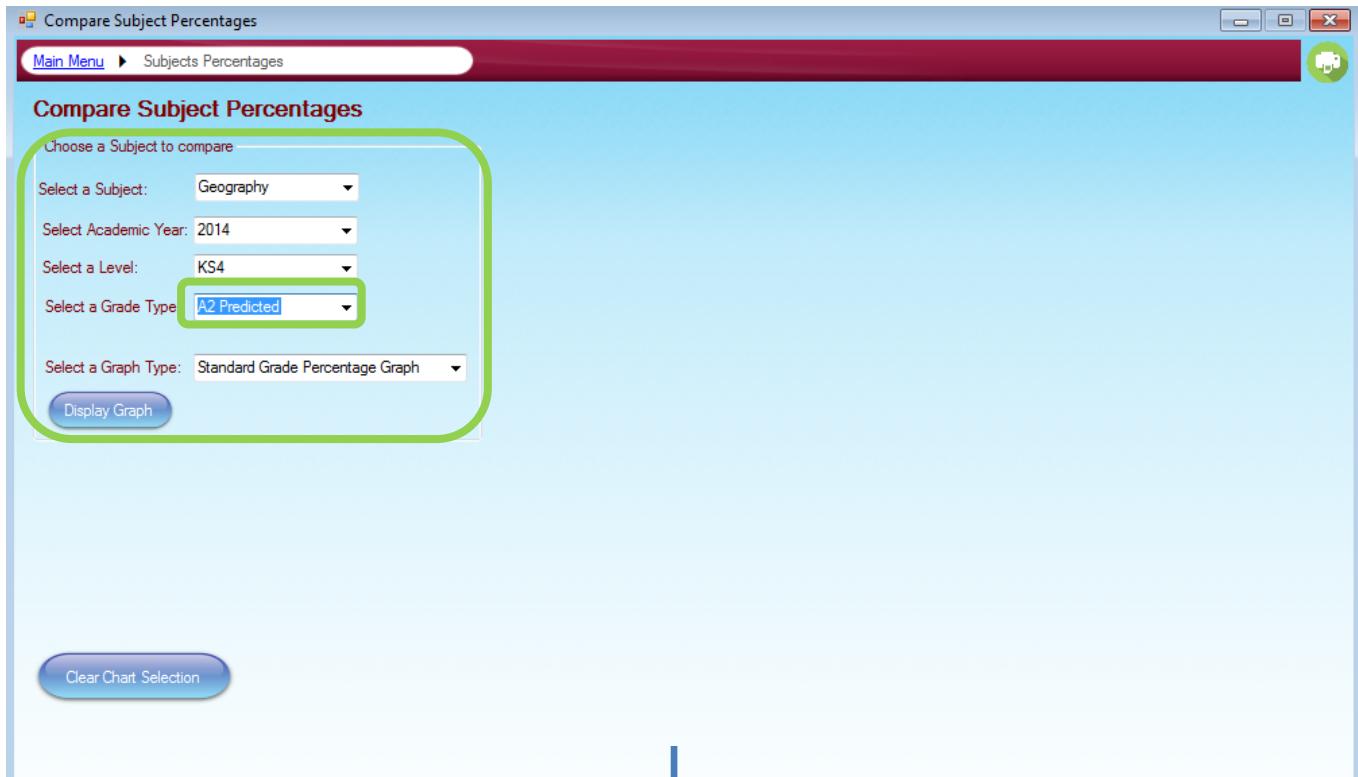
Reference 80

Evidence which demonstrates the functionality of the toggle chart labels button.



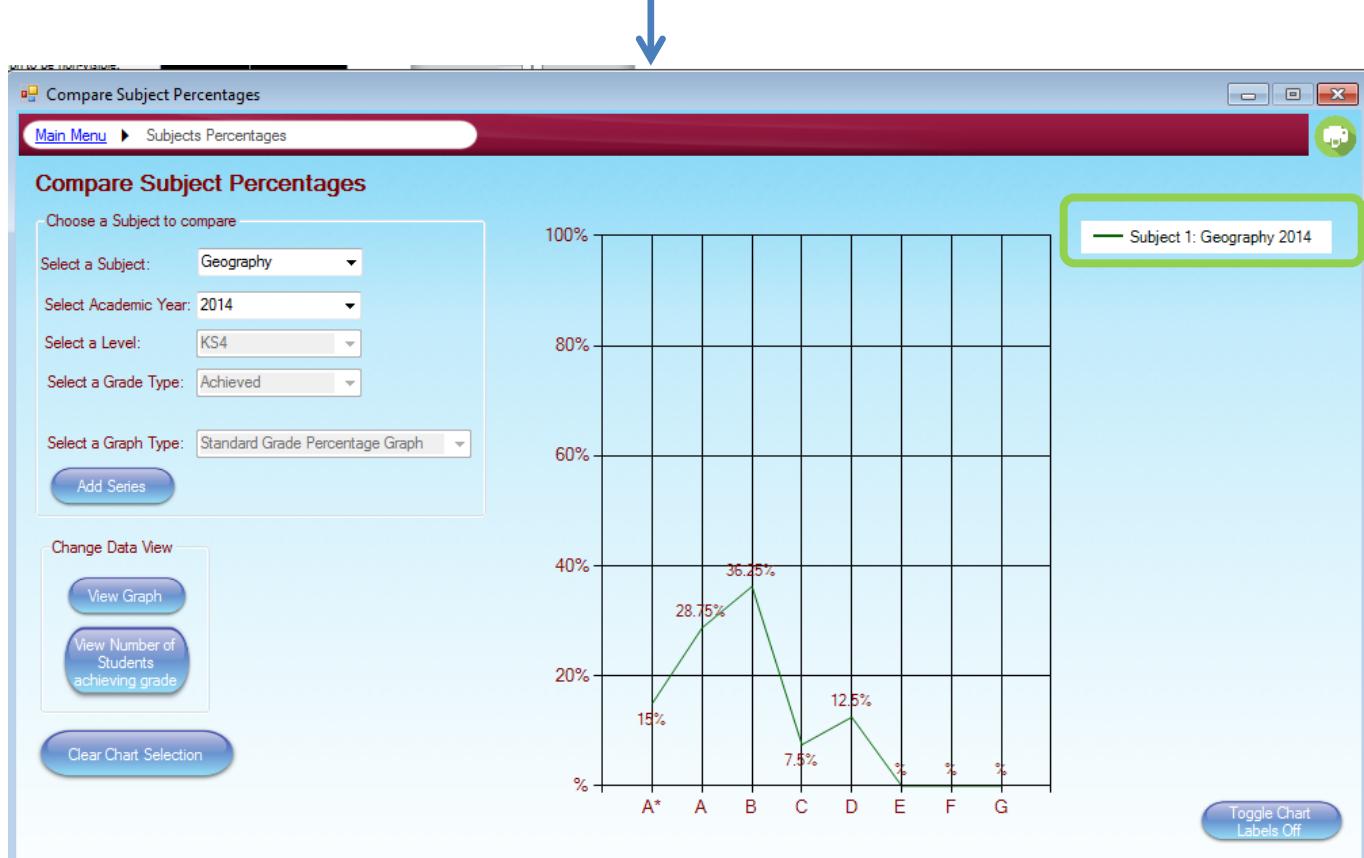
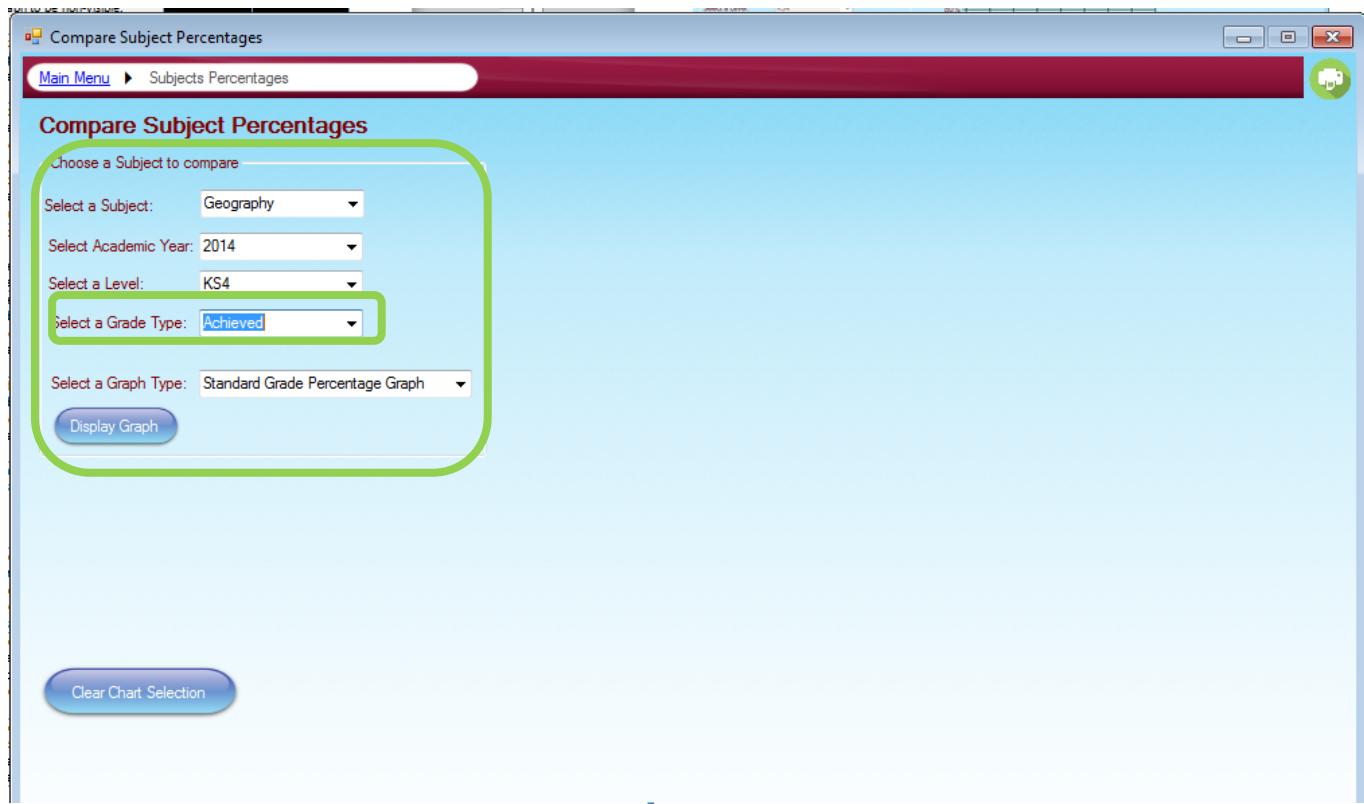
Reference 81

Evidence which shows adding a predicted grades series.



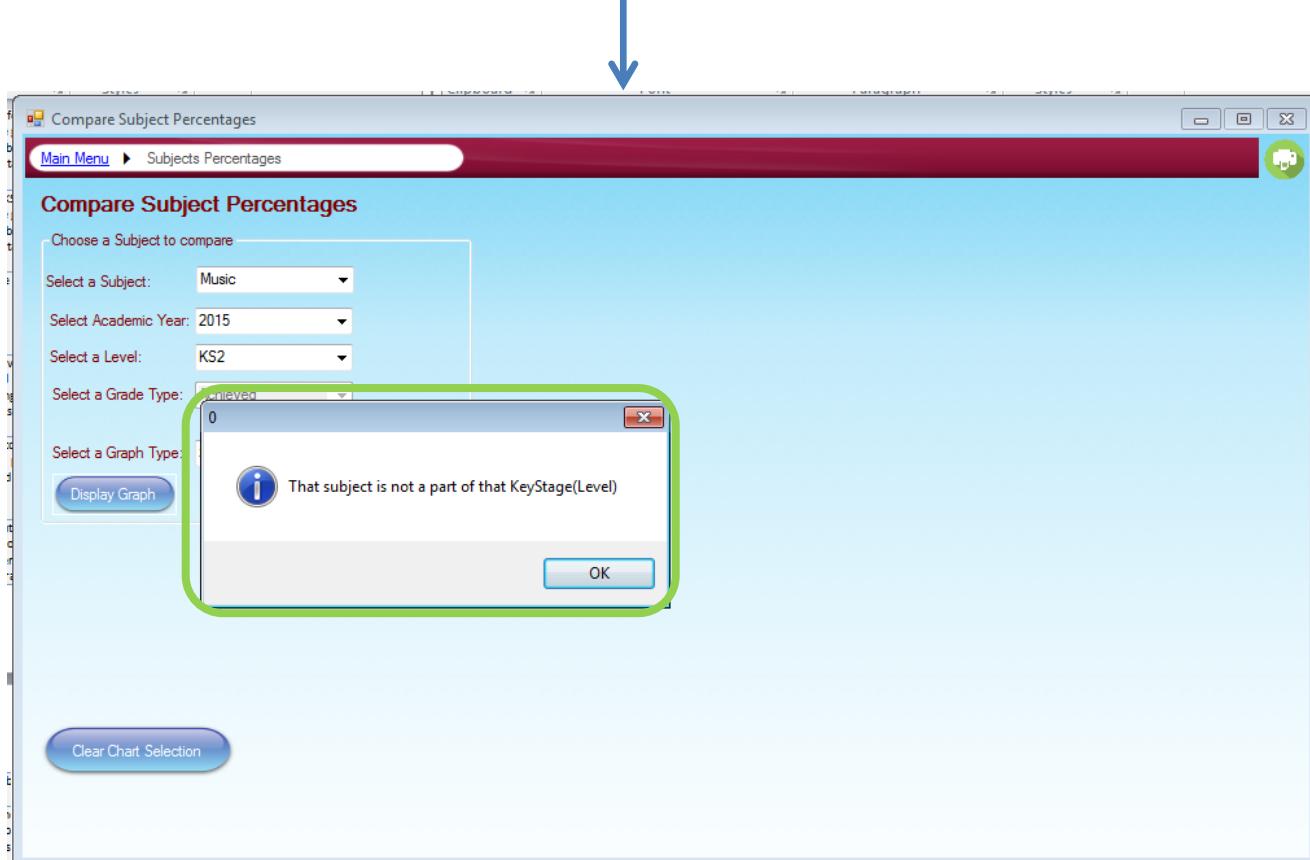
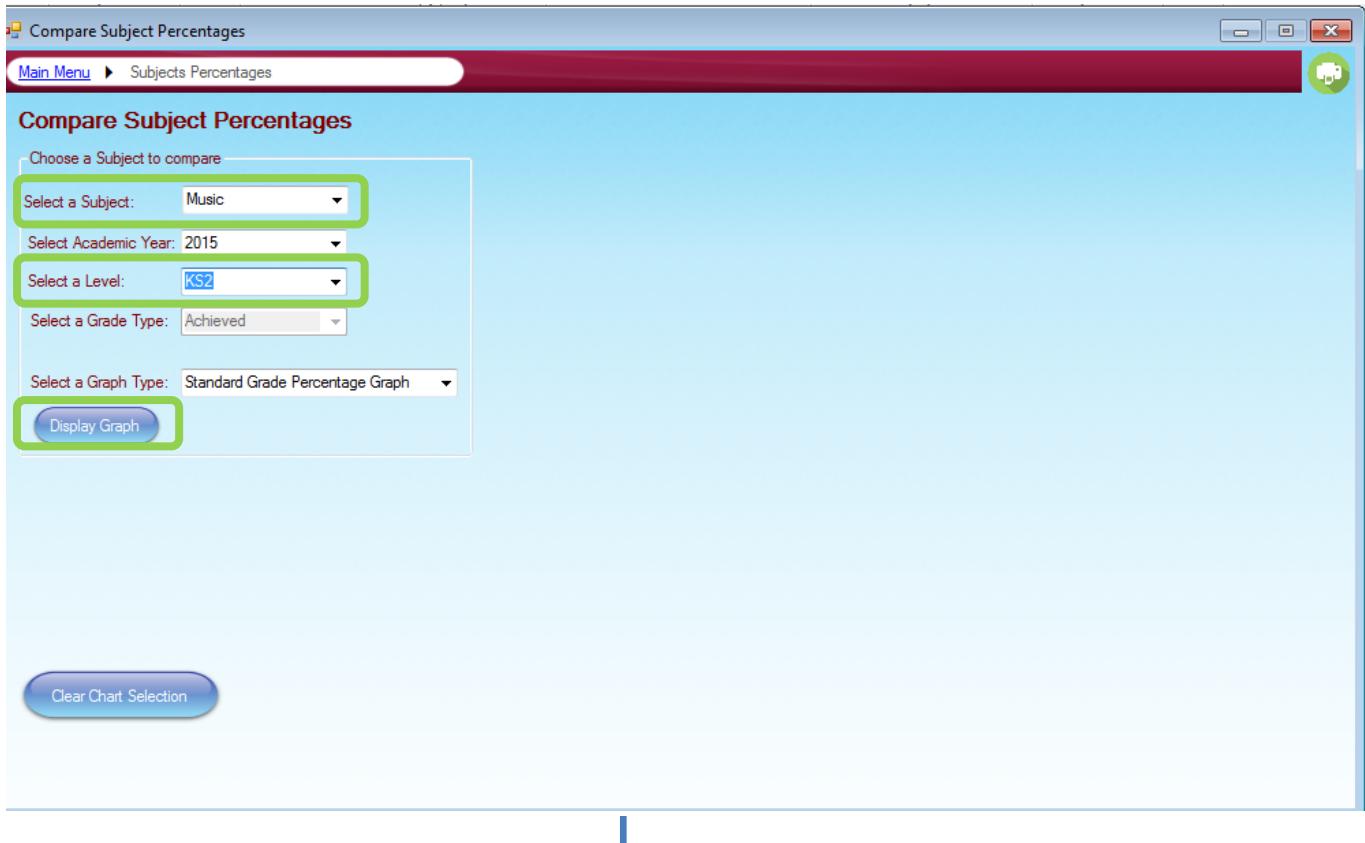
Reference 82

Evidence which shows adding an achieved grades series.



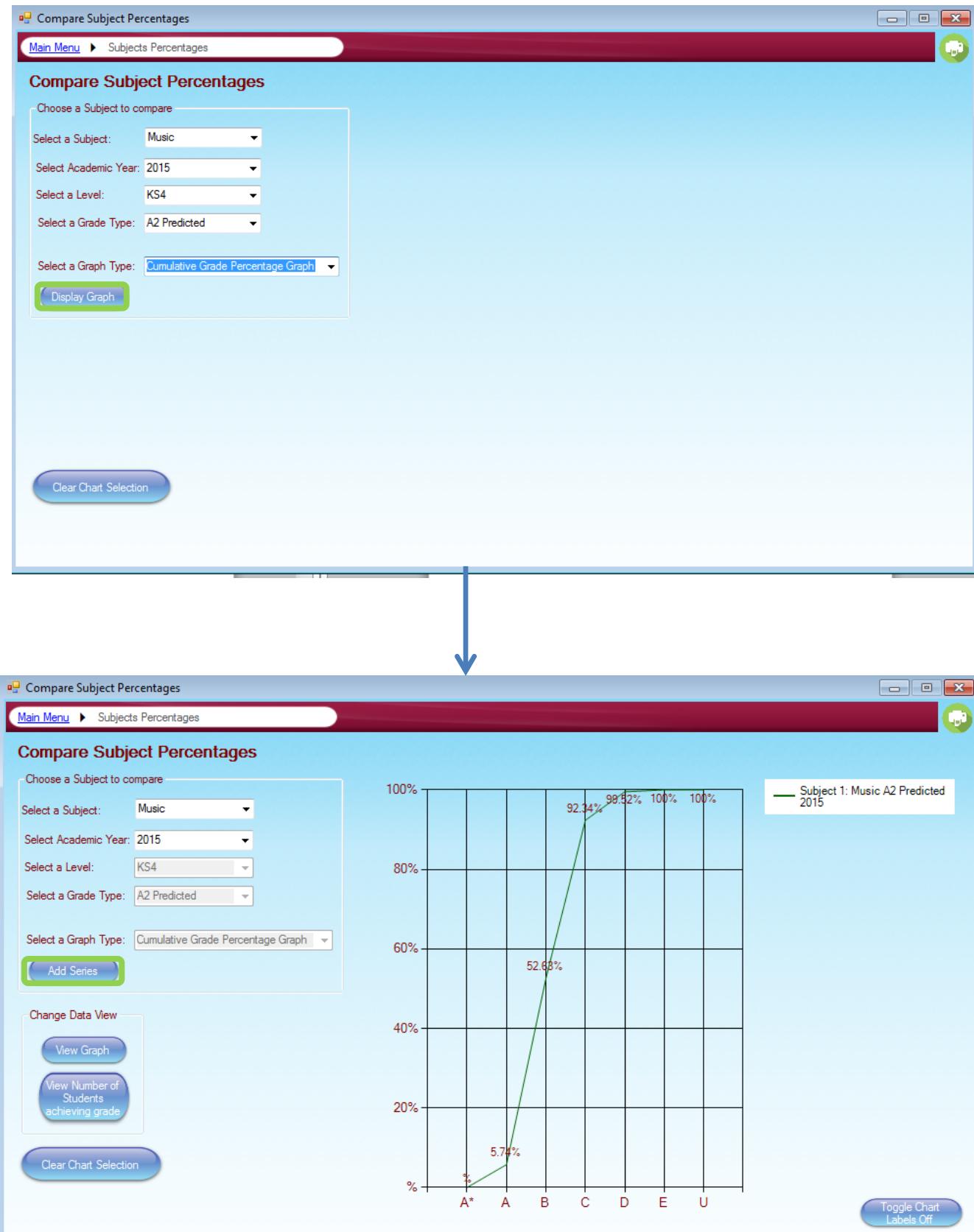
Reference 83

Evidence which shows validation within the system.



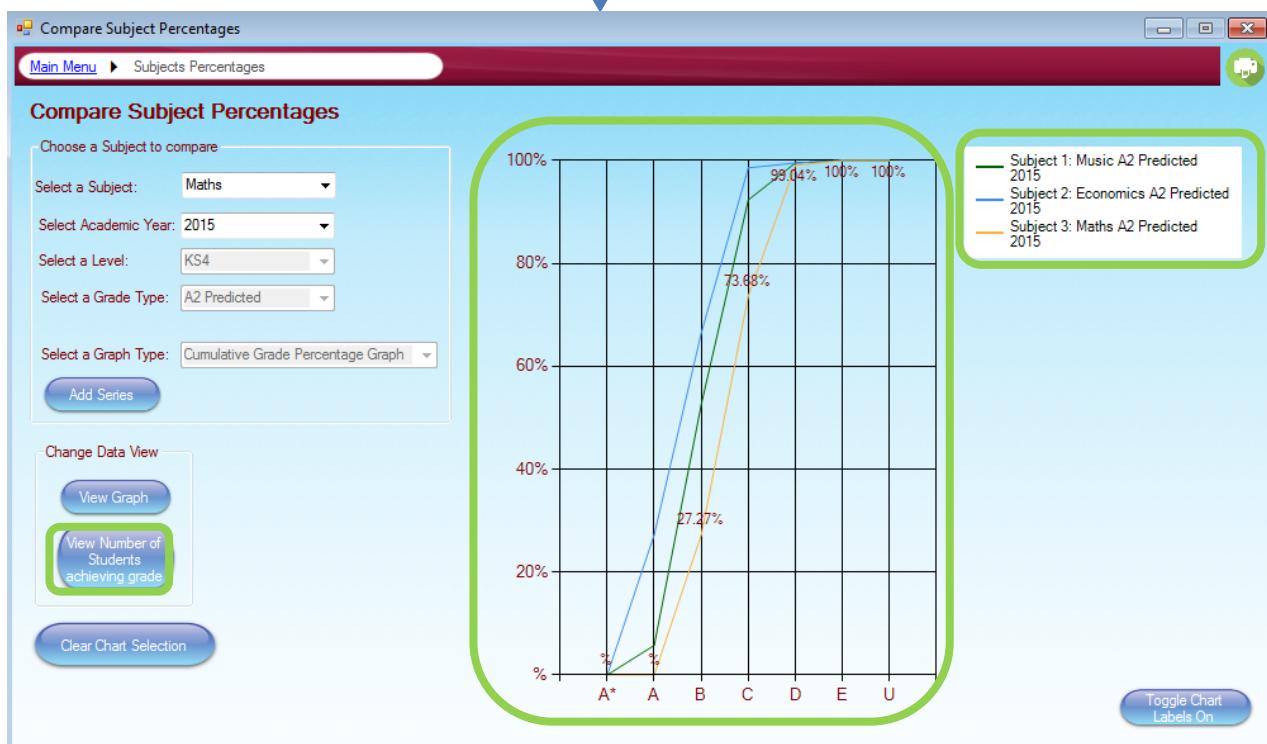
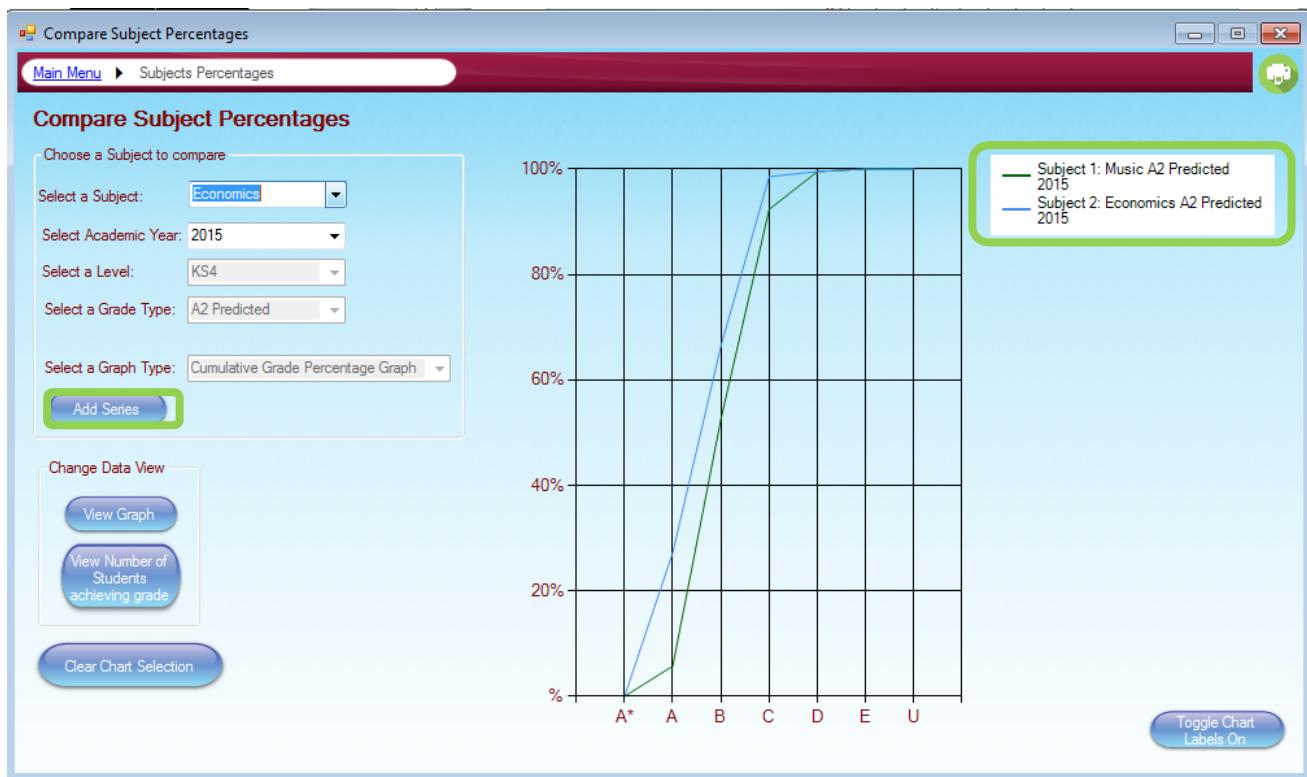
Reference 84

Evidence which shows the successful adding of one and displaying of one series.

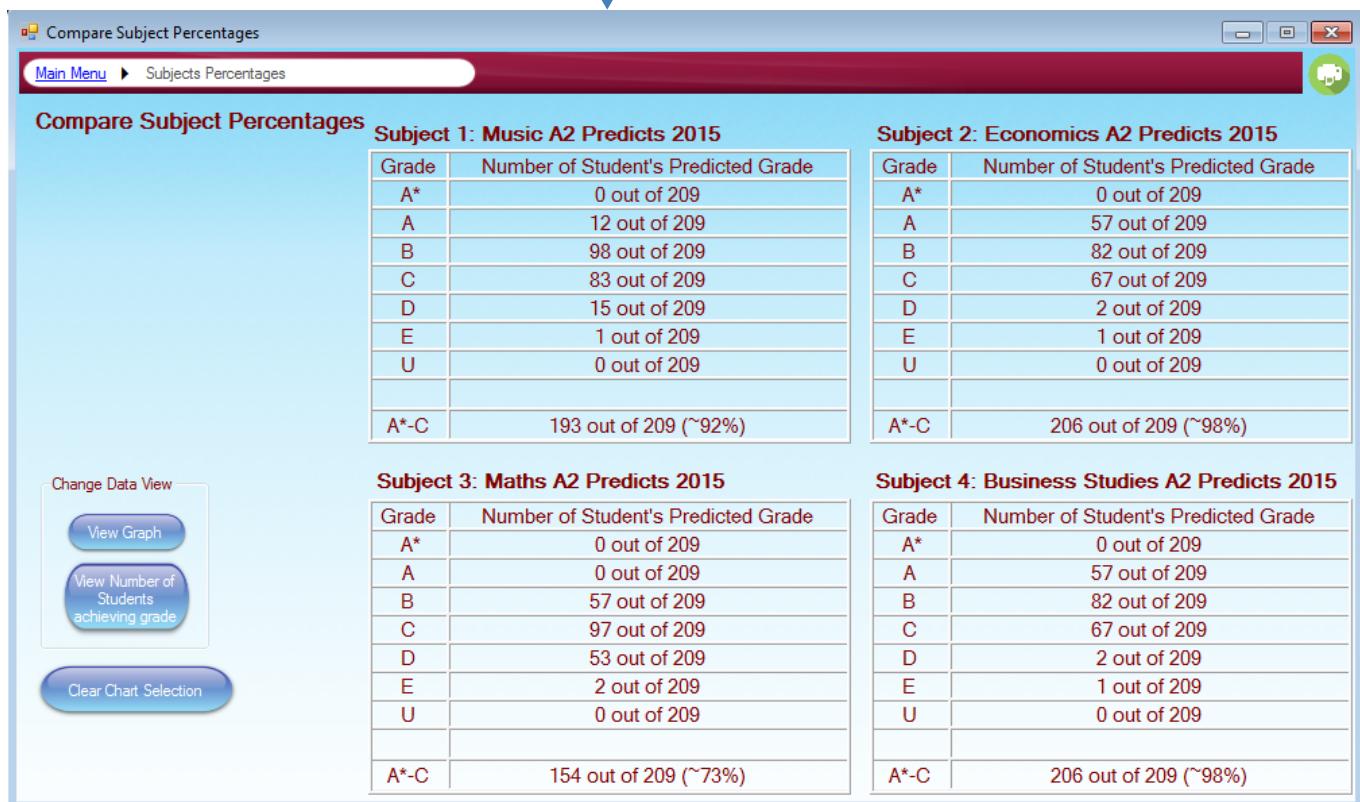


Reference 85

Evidence which demonstrates adding more than one chart series.

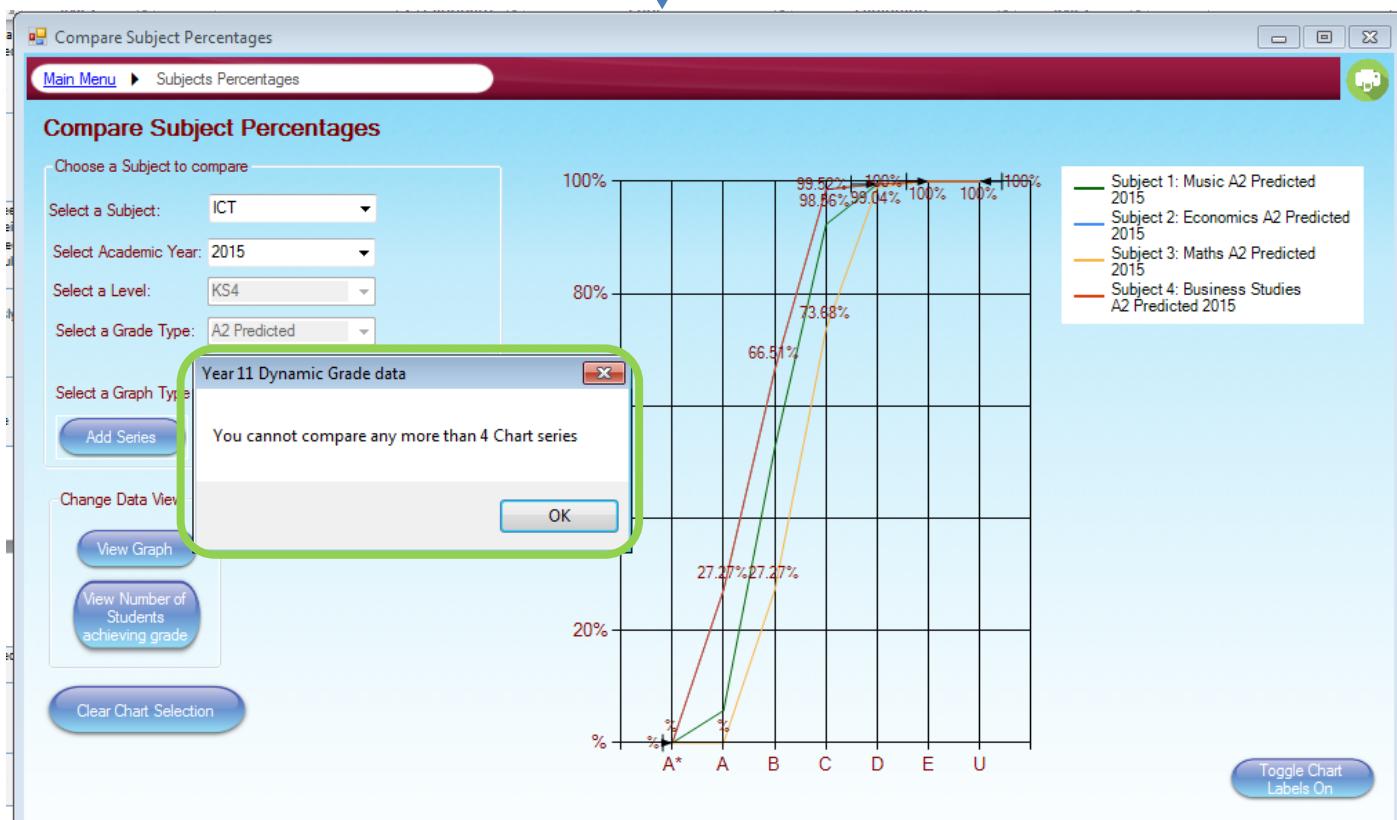
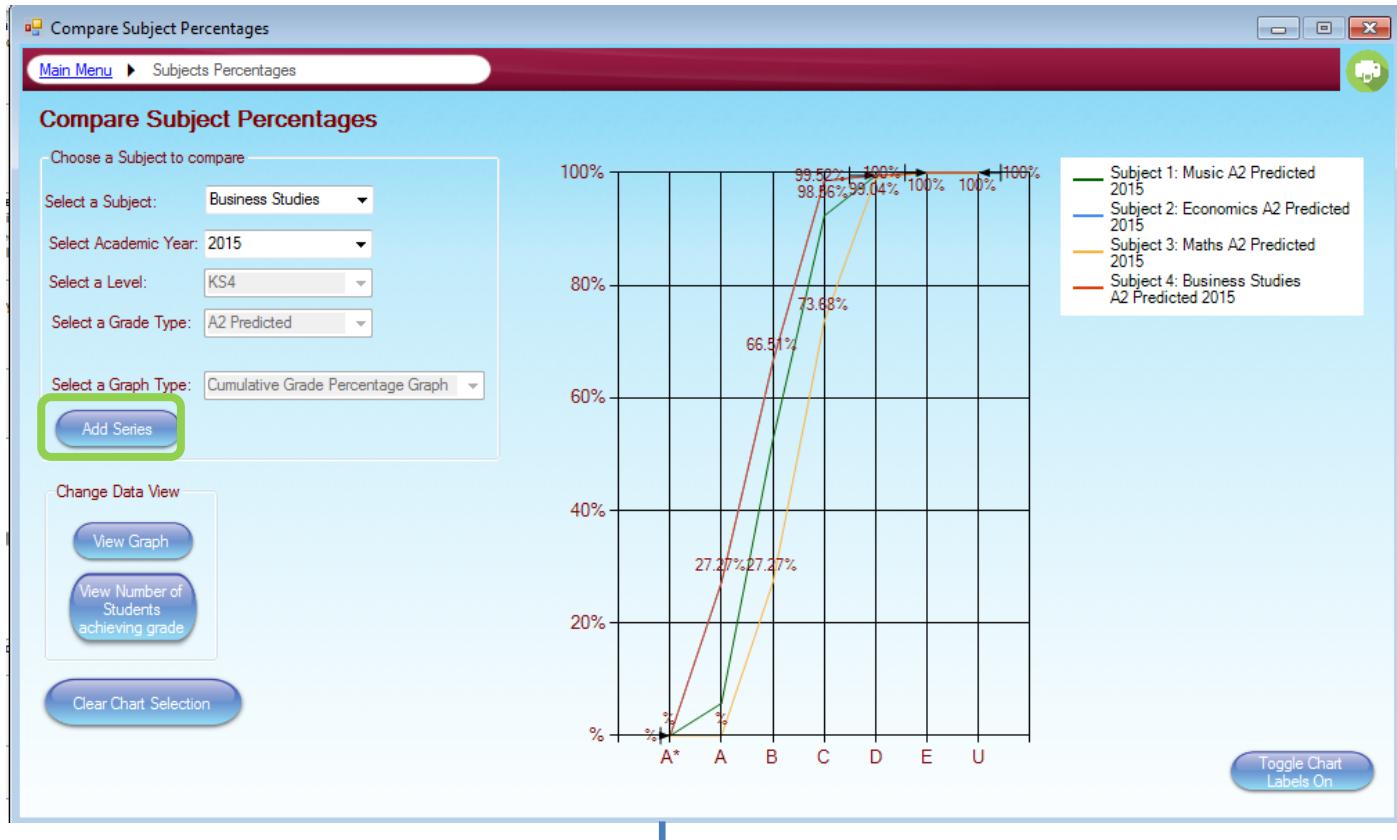


Reference 85 (continued)



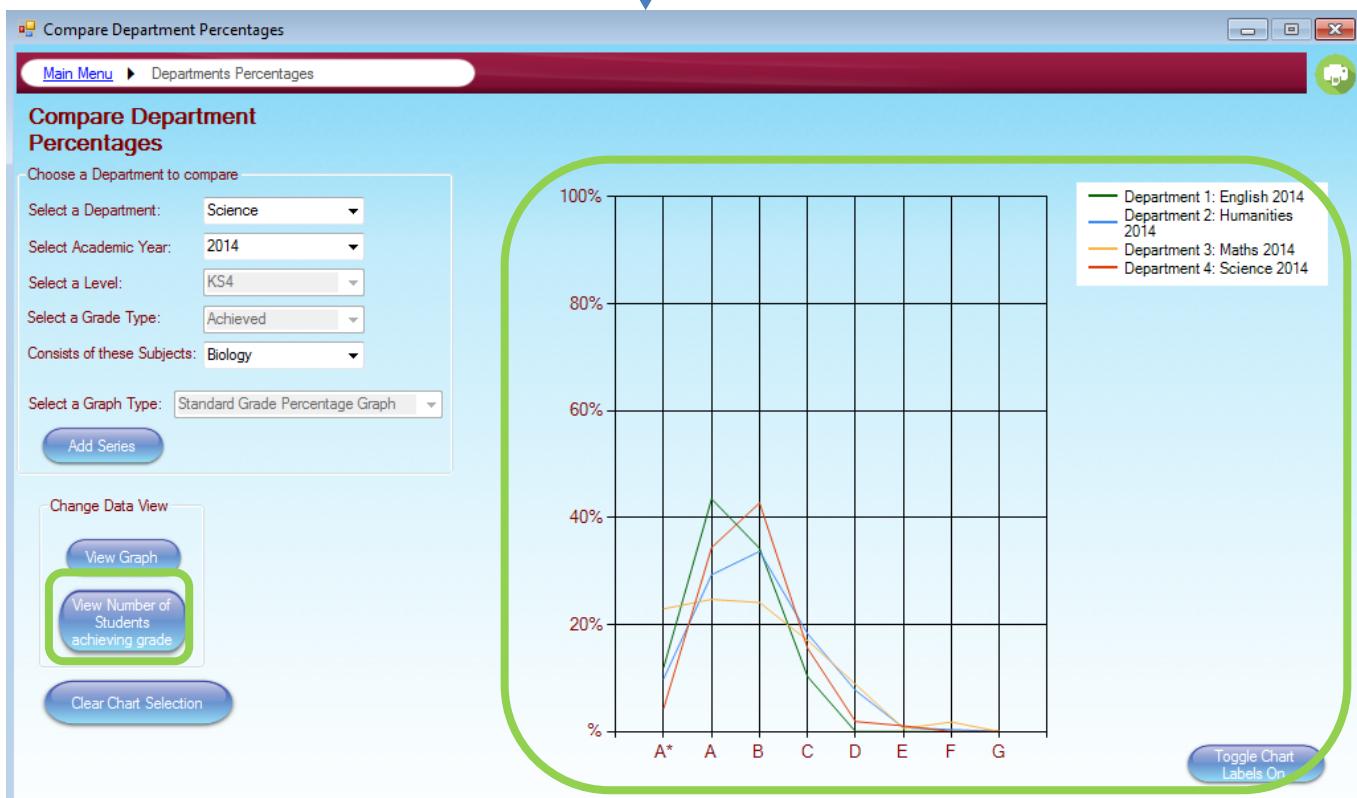
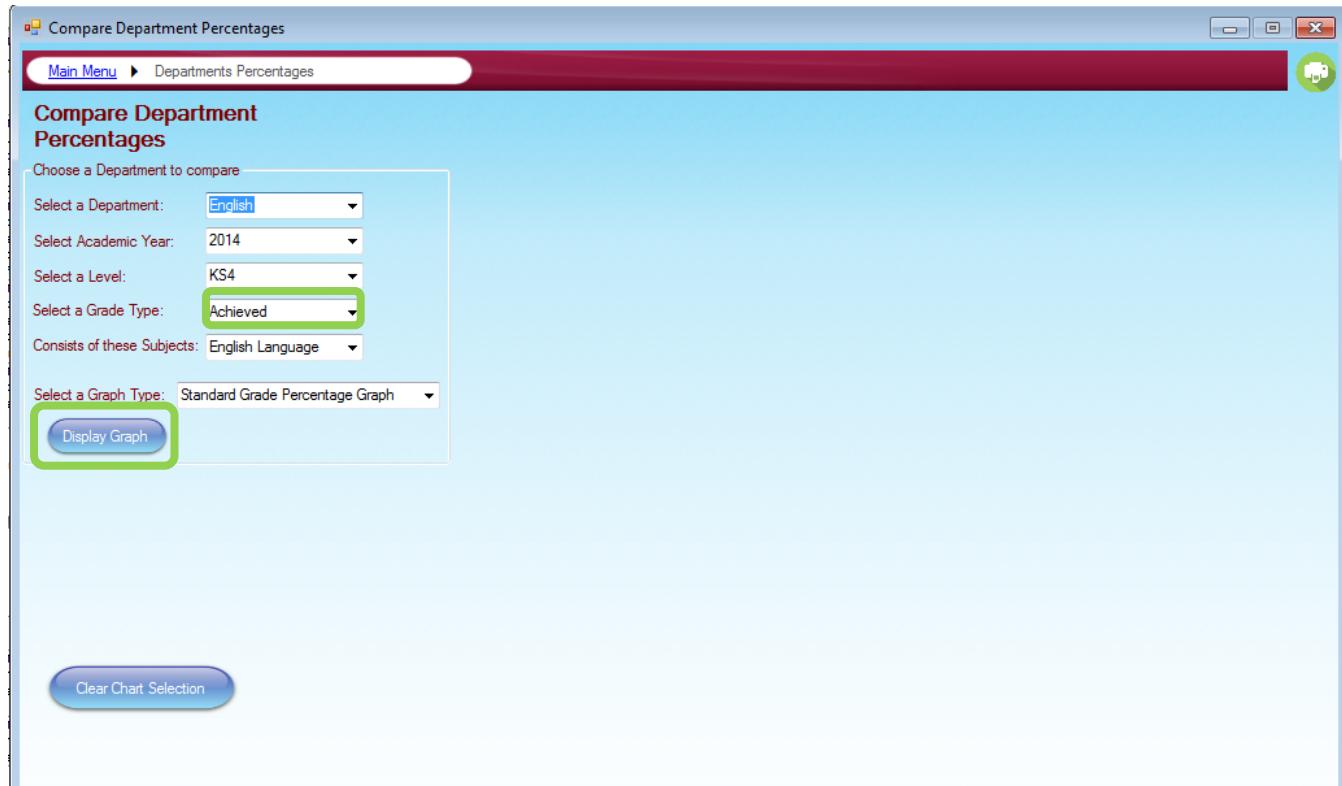
Reference 86

Evidence which demonstrates what happens when a user attempts to add more than four series.



Reference 87

This evidence shows the functionality of the comparing achieved grades on the compare department percentages form.



Reference 87 (continued)



Compare Department Percentages

Main Menu > Departments Percentages

Compare Department Percentages

Department 1: English 2014

Grade	Number of Students achieving Grade
A*	35 out of 292
A	127 out of 292
B	100 out of 292
C	30 out of 292
D	0 out of 292
E	0 out of 292
F	0 out of 292
G	0 out of 292
A*-C	292 out of 292 (~100%)

Department 2: Humanities 2014

Grade	Number of Students achieving Grade
A*	27 out of 273
A	80 out of 273
B	92 out of 273
C	50 out of 273
D	21 out of 273
E	2 out of 273
F	1 out of 273
G	0 out of 273
A*-C	249 out of 273 (~91%)

Department 3: Maths 2014

Grade	Number of Students achieving Grade
A*	39 out of 170
A	42 out of 170
B	41 out of 170
C	29 out of 170
D	15 out of 170
E	1 out of 170
F	3 out of 170
G	0 out of 170
A*-C	151 out of 170 (~88%)

Department 4: Science 2014

Grade	Number of Students achieving Grade
A*	16 out of 372
A	128 out of 372
B	159 out of 372
C	58 out of 372
D	7 out of 372
E	4 out of 372
F	0 out of 372
G	0 out of 372
A*-C	361 out of 372 (~97%)

Change Data View

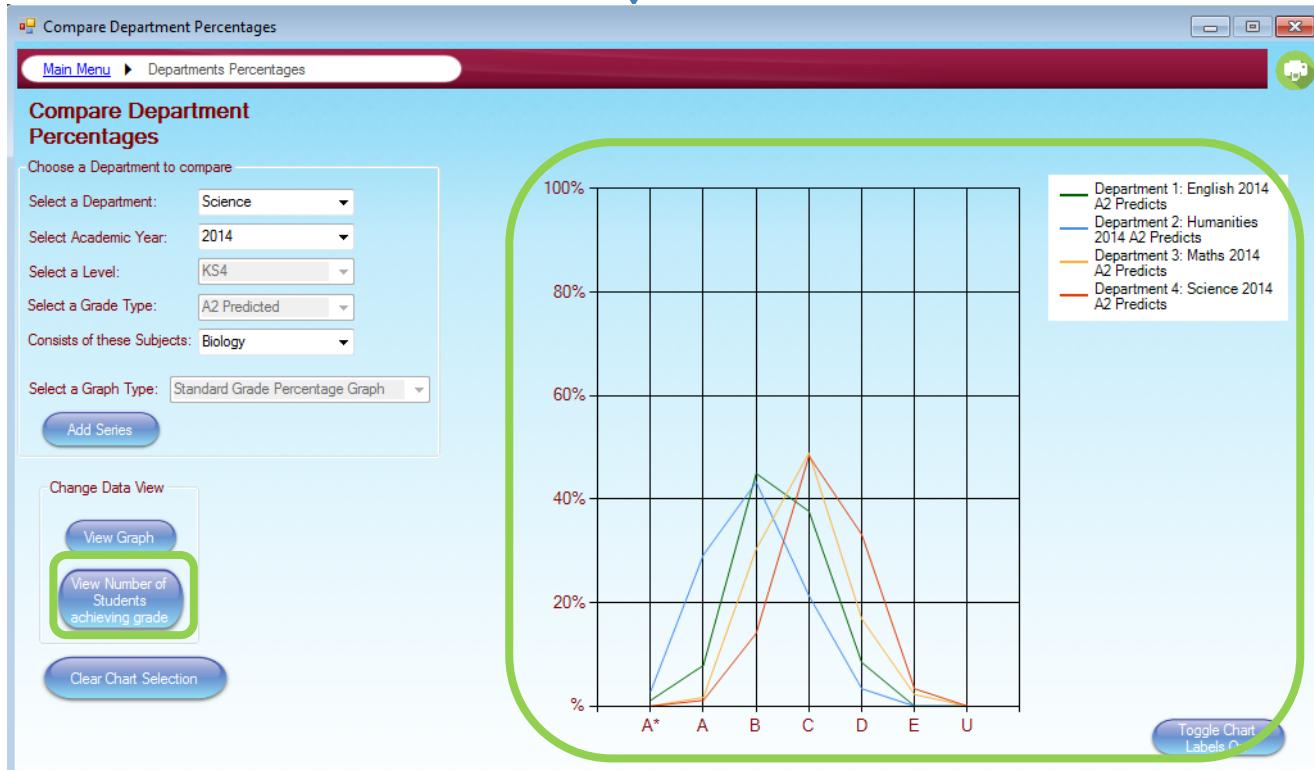
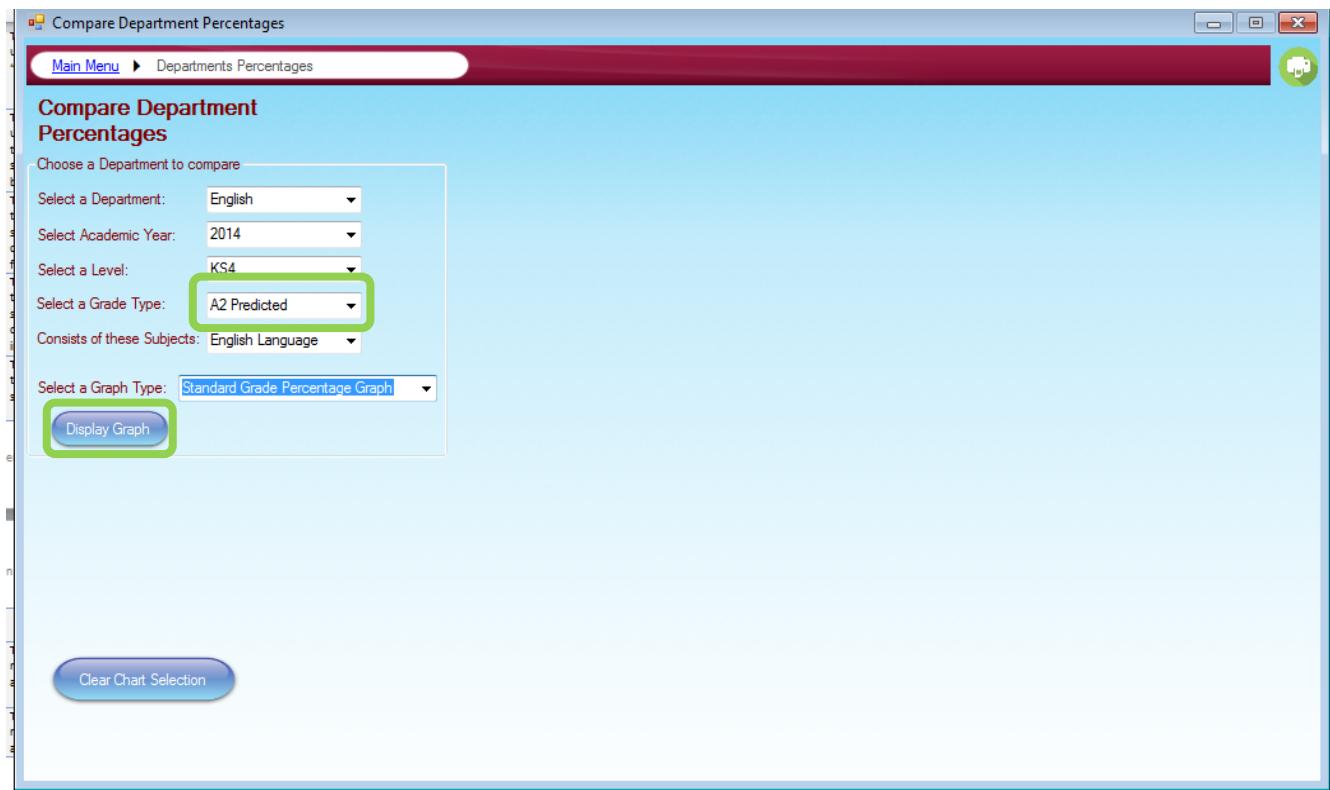
View Graph

View Number of Students achieving grade

Clear Chart Selection

Reference 88

This evidence shows the functionality of the comparing predicted grades on the compare department percentages form.



Reference 88 (continued)



Compare Department Percentages

Main Menu > Departments Percentages

Compare Department Percentages

Department 1: English 2014 A2 Predicts

Grade	Number of Student's Predicted Grade
A*	4 out of 356
A	28 out of 356
B	160 out of 356
C	134 out of 356
D	30 out of 356
E	0 out of 356
U	0 out of 356
A*-C	326 out of 356 (~91%)

Department 2: Humanities 2014 A2 Predicts

Grade	Number of Student's Predicted Grade
A*	30 out of 1068
A	312 out of 1068
B	462 out of 1068
C	228 out of 1068
D	36 out of 1068
E	0 out of 1068
U	0 out of 1068
A*-C	1032 out of 1068 (~96%)

Department 3: Maths 2014 A2 Predicts

Grade	Number of Student's Predicted Grade
A*	0 out of 178
A	3 out of 178
B	54 out of 178
C	87 out of 178
D	30 out of 178
E	4 out of 178
U	0 out of 178
A*-C	144 out of 178 (~80%)

Department 4: Science 2014 A2 Predicts

Grade	Number of Student's Predicted Grade
A*	0 out of 534
A	6 out of 534
B	75 out of 534
C	258 out of 534
D	177 out of 534
E	18 out of 534
U	0 out of 534
A*-C	339 out of 534 (~63%)

Change Data View

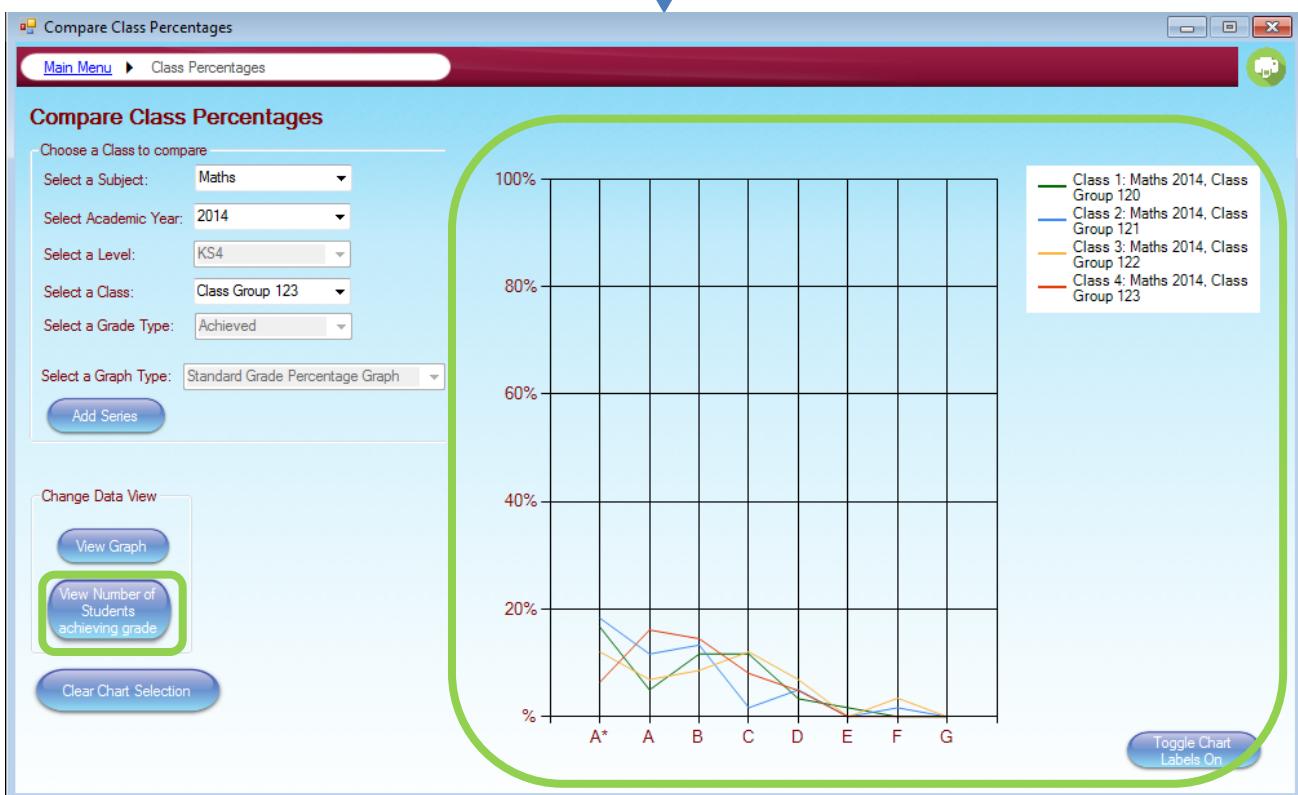
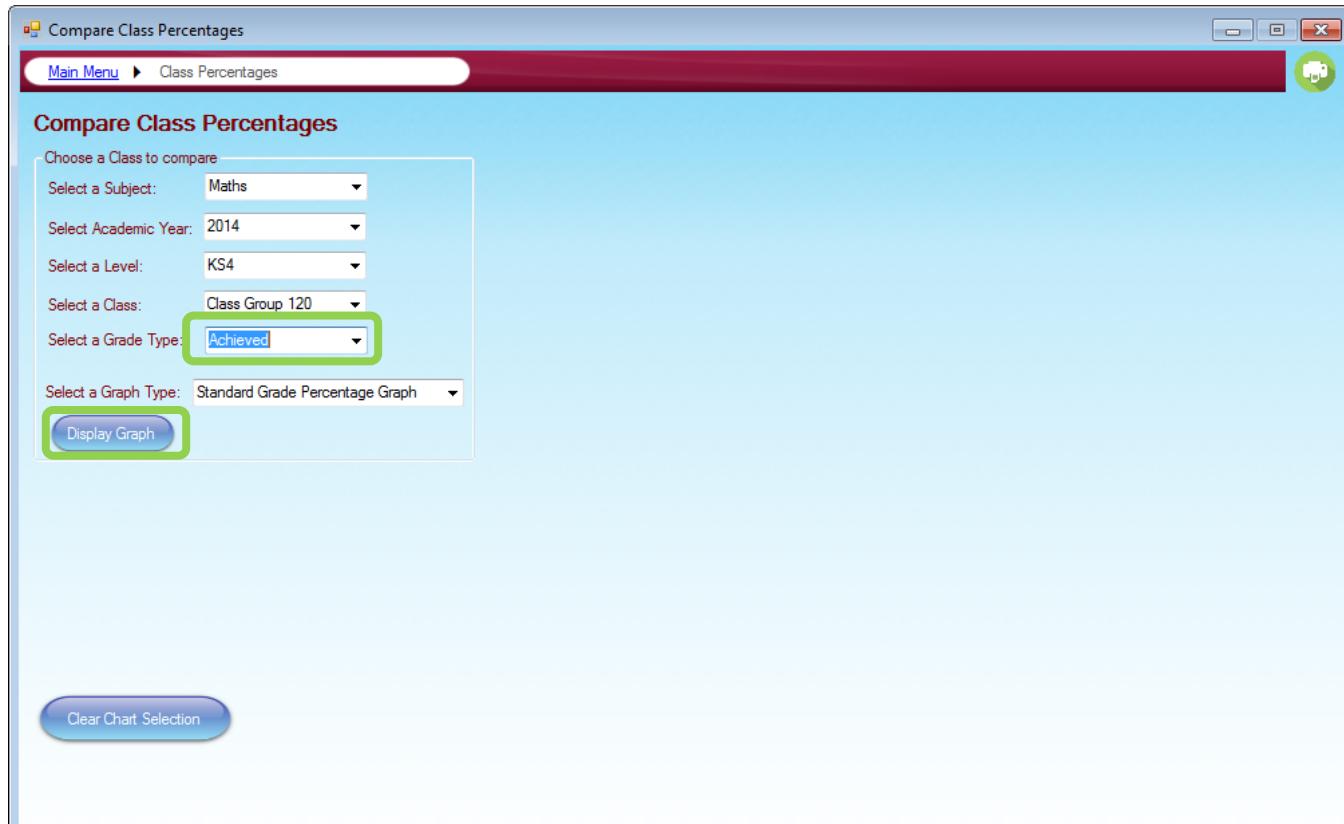
View Graph

View Number of Students achieving grade

Clear Chart Selection

Reference 89

This evidence shows the functionality of the comparing achieved grades on the compare class percentages form.



Reference 89 (continued)



Compare Class Percentages																																									
Main Menu > Class Percentages																																									
Class 1: Maths 2014, Class Group 120	Class 2: Maths 2014, Class Group 121																																								
<table border="1"><thead><tr><th>Grade</th><th>Number of Students achieving Grade</th></tr></thead><tbody><tr><td>A*</td><td>10 out of 60</td></tr><tr><td>A</td><td>3 out of 60</td></tr><tr><td>B</td><td>7 out of 60</td></tr><tr><td>C</td><td>7 out of 60</td></tr><tr><td>D</td><td>2 out of 60</td></tr><tr><td>E</td><td>1 out of 60</td></tr><tr><td>F</td><td>0 out of 60</td></tr><tr><td>G</td><td>0 out of 60</td></tr><tr><td>A*-C</td><td>27 out of 60 (~45%)</td></tr></tbody></table>	Grade	Number of Students achieving Grade	A*	10 out of 60	A	3 out of 60	B	7 out of 60	C	7 out of 60	D	2 out of 60	E	1 out of 60	F	0 out of 60	G	0 out of 60	A*-C	27 out of 60 (~45%)	<table border="1"><thead><tr><th>Grade</th><th>Number of Students achieving Grade</th></tr></thead><tbody><tr><td>A*</td><td>11 out of 60</td></tr><tr><td>A</td><td>7 out of 60</td></tr><tr><td>B</td><td>8 out of 60</td></tr><tr><td>C</td><td>1 out of 60</td></tr><tr><td>D</td><td>3 out of 60</td></tr><tr><td>E</td><td>0 out of 60</td></tr><tr><td>F</td><td>1 out of 60</td></tr><tr><td>G</td><td>0 out of 60</td></tr><tr><td>A*-C</td><td>27 out of 60 (~45%)</td></tr></tbody></table>	Grade	Number of Students achieving Grade	A*	11 out of 60	A	7 out of 60	B	8 out of 60	C	1 out of 60	D	3 out of 60	E	0 out of 60	F	1 out of 60	G	0 out of 60	A*-C	27 out of 60 (~45%)
Grade	Number of Students achieving Grade																																								
A*	10 out of 60																																								
A	3 out of 60																																								
B	7 out of 60																																								
C	7 out of 60																																								
D	2 out of 60																																								
E	1 out of 60																																								
F	0 out of 60																																								
G	0 out of 60																																								
A*-C	27 out of 60 (~45%)																																								
Grade	Number of Students achieving Grade																																								
A*	11 out of 60																																								
A	7 out of 60																																								
B	8 out of 60																																								
C	1 out of 60																																								
D	3 out of 60																																								
E	0 out of 60																																								
F	1 out of 60																																								
G	0 out of 60																																								
A*-C	27 out of 60 (~45%)																																								
Class 3: Maths 2014, Class Group 122	Class 4: Maths 2014, Class Group 123																																								
<table border="1"><thead><tr><th>Grade</th><th>Number of Students achieving Grade</th></tr></thead><tbody><tr><td>A*</td><td>7 out of 58</td></tr><tr><td>A</td><td>4 out of 58</td></tr><tr><td>B</td><td>5 out of 58</td></tr><tr><td>C</td><td>7 out of 58</td></tr><tr><td>D</td><td>4 out of 58</td></tr><tr><td>E</td><td>0 out of 58</td></tr><tr><td>F</td><td>2 out of 58</td></tr><tr><td>G</td><td>0 out of 58</td></tr><tr><td>A*-C</td><td>23 out of 58 (~39%)</td></tr></tbody></table>	Grade	Number of Students achieving Grade	A*	7 out of 58	A	4 out of 58	B	5 out of 58	C	7 out of 58	D	4 out of 58	E	0 out of 58	F	2 out of 58	G	0 out of 58	A*-C	23 out of 58 (~39%)	<table border="1"><thead><tr><th>Grade</th><th>Number of Students achieving Grade</th></tr></thead><tbody><tr><td>A*</td><td>4 out of 62</td></tr><tr><td>A</td><td>10 out of 62</td></tr><tr><td>B</td><td>9 out of 62</td></tr><tr><td>C</td><td>5 out of 62</td></tr><tr><td>D</td><td>3 out of 62</td></tr><tr><td>E</td><td>0 out of 62</td></tr><tr><td>F</td><td>0 out of 62</td></tr><tr><td>G</td><td>0 out of 62</td></tr><tr><td>A*-C</td><td>28 out of 62 (~45%)</td></tr></tbody></table>	Grade	Number of Students achieving Grade	A*	4 out of 62	A	10 out of 62	B	9 out of 62	C	5 out of 62	D	3 out of 62	E	0 out of 62	F	0 out of 62	G	0 out of 62	A*-C	28 out of 62 (~45%)
Grade	Number of Students achieving Grade																																								
A*	7 out of 58																																								
A	4 out of 58																																								
B	5 out of 58																																								
C	7 out of 58																																								
D	4 out of 58																																								
E	0 out of 58																																								
F	2 out of 58																																								
G	0 out of 58																																								
A*-C	23 out of 58 (~39%)																																								
Grade	Number of Students achieving Grade																																								
A*	4 out of 62																																								
A	10 out of 62																																								
B	9 out of 62																																								
C	5 out of 62																																								
D	3 out of 62																																								
E	0 out of 62																																								
F	0 out of 62																																								
G	0 out of 62																																								
A*-C	28 out of 62 (~45%)																																								
Change Data View																																									
View Graph																																									
View Number of Students achieving grade																																									
Clear Chart Selection																																									

Reference 90

This evidence shows the functionality of the comparing predicted grades on the compare class percentages form.

Compare Class Percentages

Main Menu > Class Percentages

Compare Class Percentages

Choose a Class to compare

Select a Subject: ICT

Select Academic Year: 2014

Select a Level: KS4

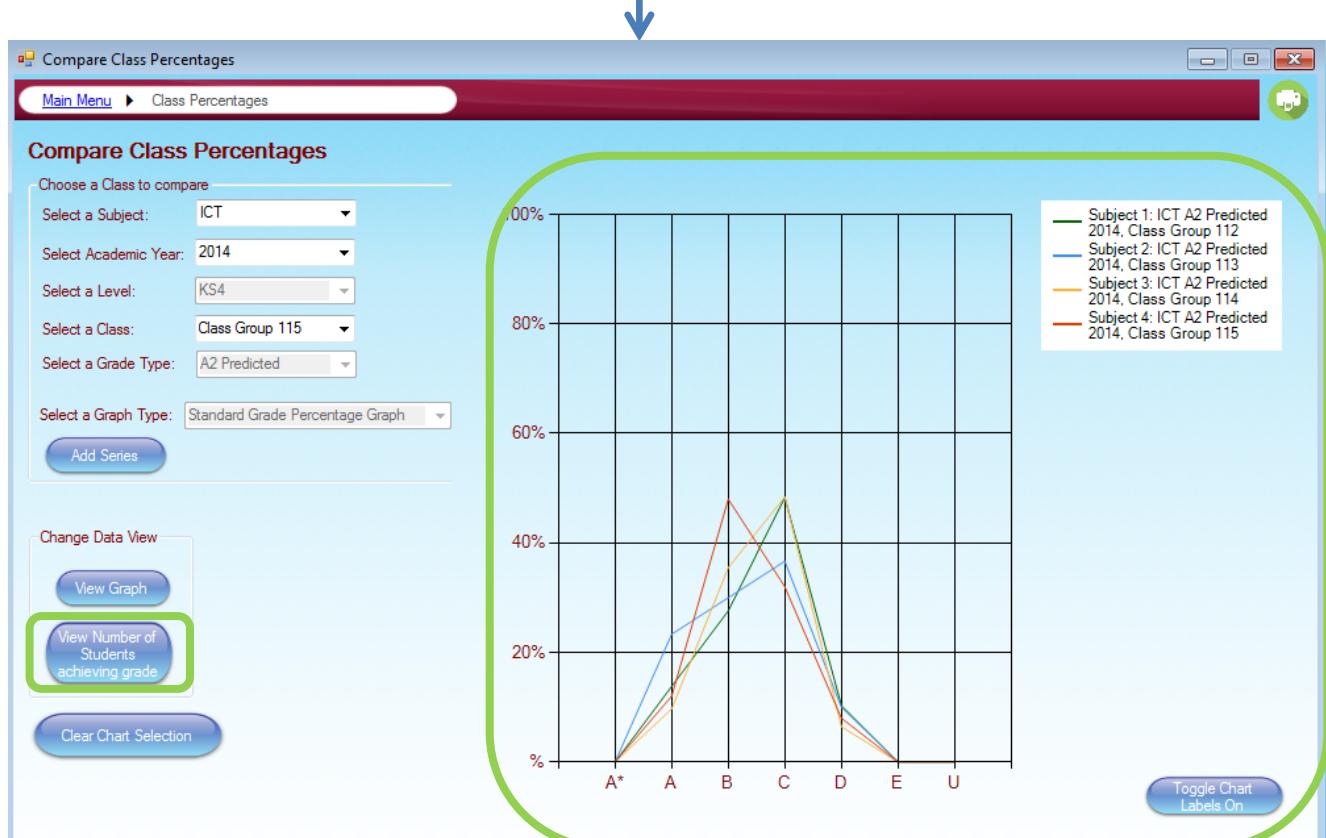
Select a Class: Class Group 112

Select a Grade Type: A2 Predicted

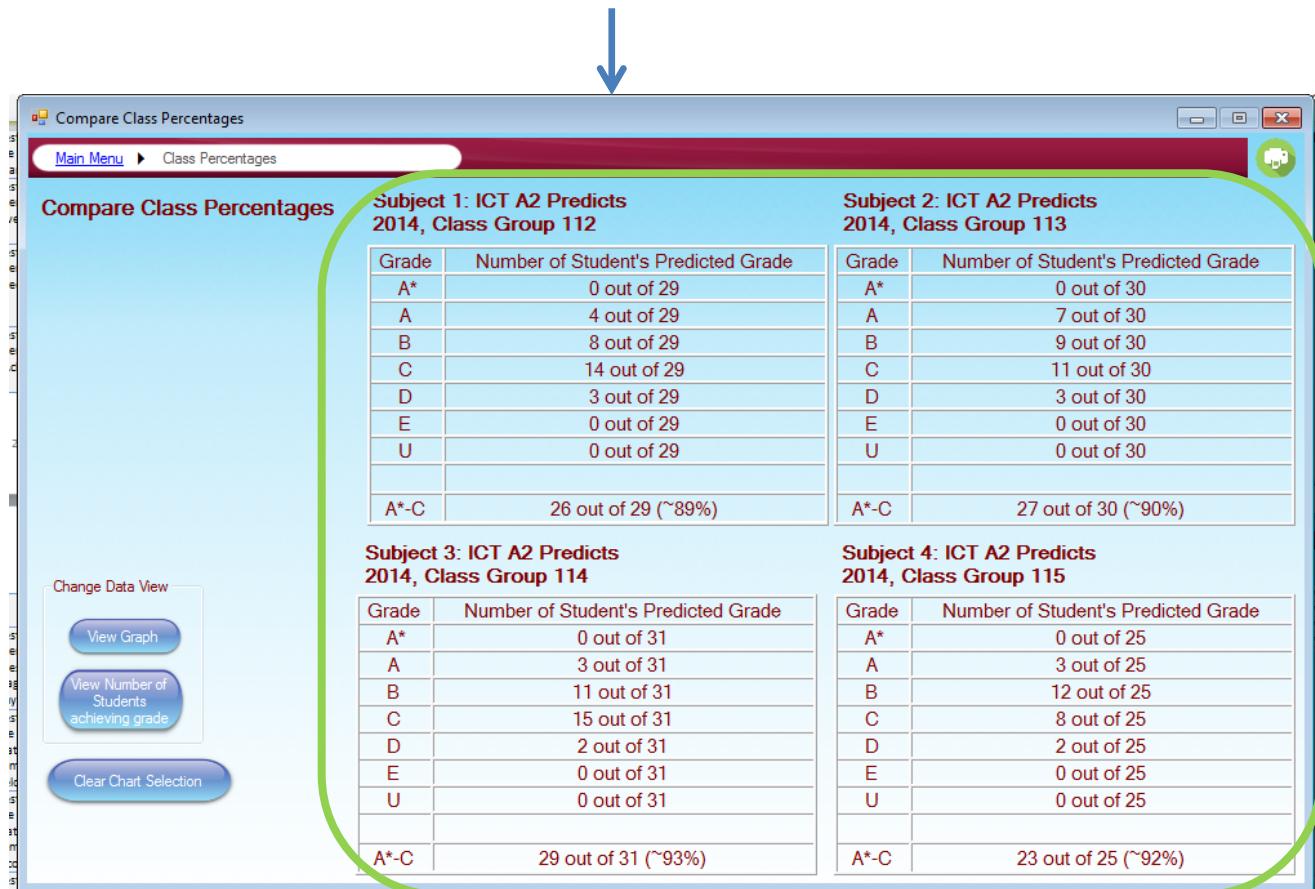
Select a Graph Type: Standard Grade Percentage Graph

Display Graph

Clear Chart Selection

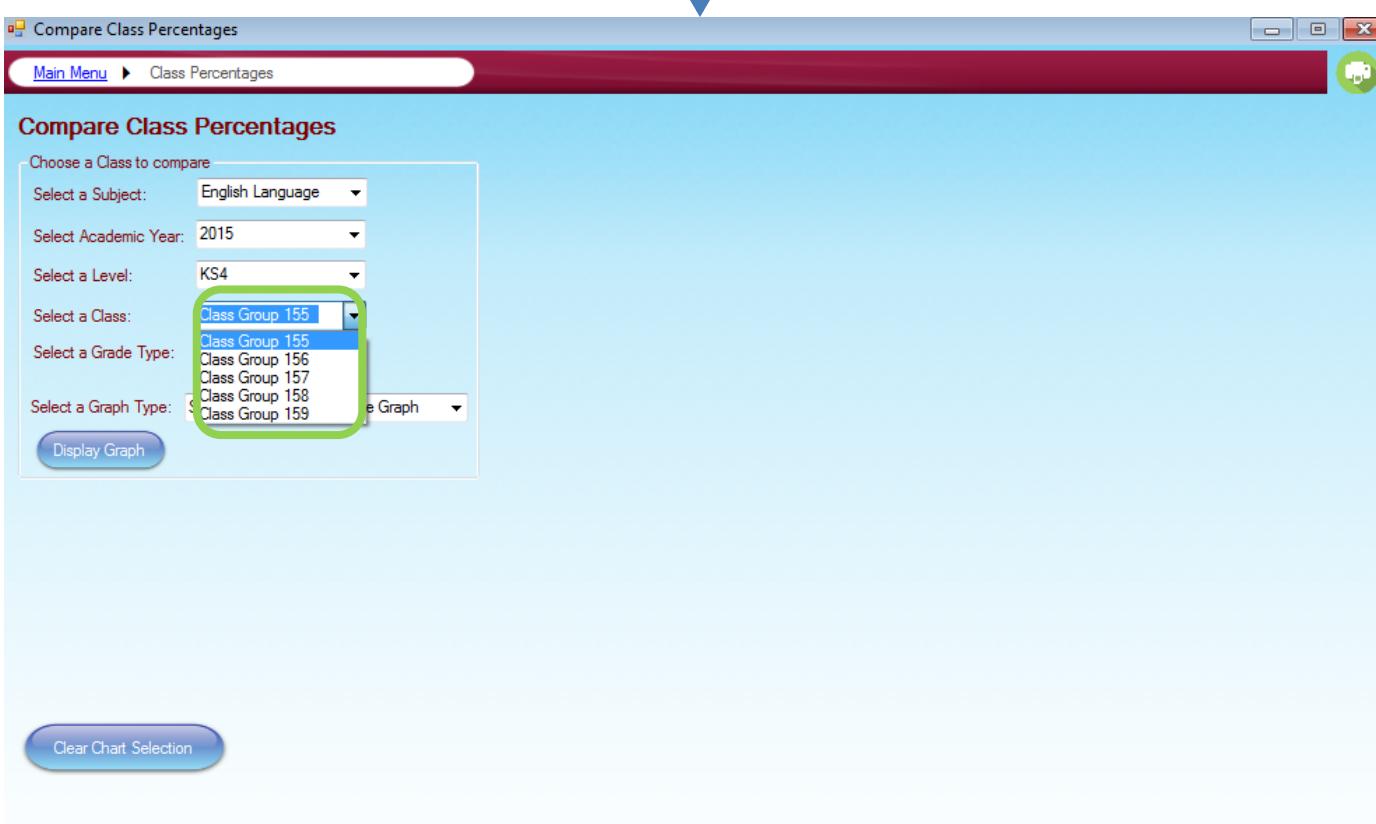
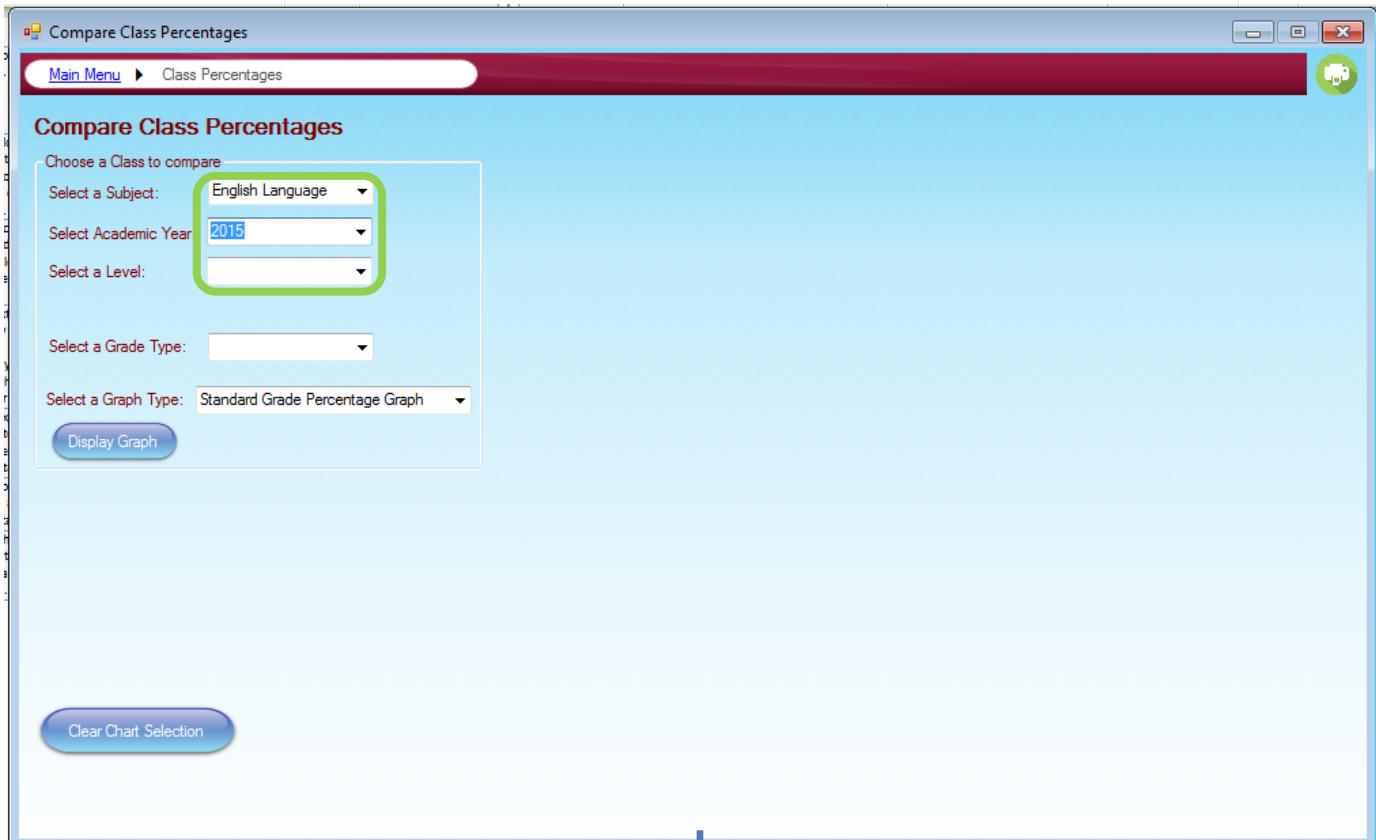


Reference 90 (continued)



Reference 91

Evidence which shows dynamic combo box generation.



Appraisal

Comparison of Project Performance against Objectives

The table below compares the performance of my completed system against the original system SMART objective targets laid out in my analysis section to assess whether objective have been met successfully or not. The following key also outlines how effective I believe my solution has been in meeting the proposed objectives.

KEY:

Objective not met	Objective partially met	Objective met	Objective Exceeded
-------------------	-------------------------	---------------	--------------------

Original objective	Completed System	Self- Evaluation of Achieving Objective
The system must be able to store data records for student details, class details, grade details and teacher details.	All data that is processed by my system is stored within a normalised set of relations within a database that is stored in a SQL server, no record is repeated more than once and there are no non-key partial dependencies.	I thought that I tackled this issue very well by designing and implementing a fully normalised set of tables within my database. I also thought that I used meaningful identifiers for field and table names.
Student details must be able to be filtered and then an administrator must be able to select a student record for editing or deleting.	Student details can be filtered using several different filter options successfully and a student's record can be edited or deleted by clicking the cell reference of a record the user wants to edit.	When it comes to editing students it wasn't very easy because I had to validate as many errors that I could think of surrounding editing and deleting data. Overall I think I did an ok job of implementing validations but there are more I could have implemented.
An administrator must be able to import large amounts of student details into the system relatively easily.	An administrative user can import student details into the system using the formatted CSV file available to them. The records that have been imported are then displayed and the user is notified if rows have been successfully added or not.	I thought that I provided a very good way in which users could import large amounts of student records. One extra thing I could add for future reference would be row formatting and conditioning to indicate errors on data importing.
Records concerning the details of teacher's should be stored and an administrator should be able to edit an existing teacher's record or add a new teacher to the system.	Teacher details records are also stored within the normalised database relations in a SQL server and the user can select to edit a teacher or add a new teacher to the system.	This area of the system was not as well developed as other areas. A user can edit and delete teachers etc., but there was a lack of teacher details stored within the database.
Class records should be stored within the system and should be identified by their department and the subject the class is concerned	Class records that are stored within the SQL database have composite keys so that they pull information from other tables about the	I think I made a very good job if implementing my database structure.

with along with the teacher that teaches the class.	teacher that teaches the class and the subject the class is a part of etc.	
Administrators should be able to add new classes to the system and edit existing classes within the system.	Only admin users of the system have access to class features which allow them to edit class details and add class details within the system.	I managed to make editing classes and adding new classes to the system very easy by using simple designs for my forms. However I could consider more validation for the future.
An administrator should be able to view all of a student's classes and add or remove students from classes.	Only admin users of the system have access to class features which allow them to add and remove students from selected classes.	Again, I thought this was implemented well but more validation for the future could be applied.
A login screen should be implemented which differentiates between admin user's and standard teacher user's.	The user login form checks a user's login record to see that if there is a 1 or a 0 in their login record within the systems database to determine whether a user has admin or non-admin privileges.	I was extremely happy with my finished login functionality, initially I had problems storing a variable which indicated whether a user was admin or not and whether to show admin or non-admin features. Although I did manage to solve this issue by storing a unique variable in my module and using succinct validation for the main menu.
There should be a separate viewing mode for admin user's and non-admin users.	The main menu form alters depending on whether an admin or non admin is logged into the system.	I managed to successfully create two separate viewing modes for both non admin and admin users using the same form. I thought I did this very well.
An admin user should be able to use important features such as editing logins, adding new subjects to the system or new academic years to the system.	An admin user has an extensive amount of extra privileges and functionalities that they can use to maintain and control the system quite easily.	An admin user of the system clearly requires a lot more responsibility than a non-admin user so I had to think of several advanced functionalities that an administrator of the system would expect to have. I managed to identify a number of different functionalities and then implement them accordingly.
An administrator should be able to filter any grade within the system.	An administrator user is able to filter all grades stored within the SQL database server by using cumulative filter selections.	I was satisfied with the result of my cumulative filter selections however one thing I could have done differently was have a single field for name filtering rather than separate first name and surname filters. This is because across my system I have used name as a filter and so in this case there is not continuity across my system.

Grade statistics of individual subject's performance's for achieved or predicted grades should be viewed graphically along with their raw data statistics.	Any user within the system can view grade percentages and statistics for any subject within the system from any academic year. Raw data statistics are also included in an alternate tabled format.	I am proud of my accomplishments regarding this area of my system because it was difficult to implement these features but I have done so in a very effective manner.
Any user should be able to view predicted grades and filter this via student, year, class or form group.	All users can access the grade predictions form which allows filtering A2 grade predictions using the following filters, student, year, class and form.	Yes this was achieved successfully.
Predicted grade data should be able to be exported from the system.	A user can generate a spread sheet CSV file of predicted grades that can be exported from the system for further use.	This was another feature of my system that adds a large scope of functionality to my system and greatly increases its usefulness.
A user should be able to logout from the system.	Users can logout from the system by clicking the logout icon on the main menu with ease.	N\A
A user should be able to produce hard-copies of important data binding forms.	Hard copies of data binding forms can be produced but I also wanted to allow a user to print only a graph selection from certain forms which I was unable to achieve.	In regards to printing forms I managed to achieve that but I never managed to achieve the printing of separate graphs.
Any user should be able to compare several series of data based on subject, class or department and this should be done for both achieved grades stored within the system and predicted grades.	Any user can select and compare classes, subjects or departments for achieved and predicted grades in the system with up to 4 series. An alternate raw statistics view is also available for viewing.	Again this was a very difficult section of my system to implement and I thought I did this very well. One slight thing that could be changed is the allowing of comparing achieved vs. predicted grades.
There must be a model that deals with generating predicted grades.	I managed to create a grade predictions model that is dynamic and creates an average point score for an individual student based on the mean relative difficulty score of their achieved grades. Then I used the individual average point score for each student to generate predicted grades for every A2 subject depending on their perceived relative difficulty ratings.	I created an extremely clever grade predictions model with subject weightings, and managed to implement this model into my system for every generated predicted grade.
A non-admin user (teacher) should be able to edit grades of students only within their own classes.	A teaching staff user can only view students that are a part of the classes they teach so they can only edit grades of students within their own classes.	Yes this was achieved successfully.
A non-admin user (teacher) should be able to see a student's grade that is within their class to see how they may be performing	A teaching staff user can only view students that are a part of the classes they teach so they can only view all the grades of an individual	Yes this was achieved successfully.

elsewhere.	student that is in their class to see how they have performed in other subjects.	
Be able to represent grades in a format that shows the number of grades that were between A* and C.	In my raw statistic tables I have created a label object that shows the number and percentage of students achieving grades A* to C.	Yes this was achieved successfully.
The system must be user friendly and easy to use.	The system was designed with usability in mind and has been implemented with even better features to allow a user to use the system with ease-of-use.	I feel like I did a great job with my designing of the system, I followed my initial designs and prototypes of the system as best I could but I also changed and added features for the better.
An administrator should be able to edit any existing grade within the system and insert any new grade for a selected student into the system.	Administrators have access to a separate grade adding form whereby they can select a student and then a new grade for the selected student. Since an admin can filter any grade within the system they can also edit any grade within the system.	I implemented lots of features which made it easy to find grades within the system and then edit them if need be or delete them. I believe my system deals with grades very well.

User Feedback

User feedback from Mr Hewitt, a primary user and secondary user of the system.

User Feedback Form

Please rate the following aspects of the fully implemented system out of five:

	5: Unusable	4: Poor	3: Neutral	2: Good	1: Excellent
The User Interface (GUI)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Adding and editing Records within the system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Comparing Achieved and Predicted Grades	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Importing Student data to the system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Exporting Predicted grades from the system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Filtering Records within the system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Additional feedback on features such as database records structure, ease of use of the system, user privileges and data security etc.

forms are well presented & visual, but message boxes are standard VB and would benefit from being in the same format as main forms. Some buttons can be clicked on multiple times and will not do anything until other options are changed. - Adding a message to these would aid clarity. When selecting Staff you can only choose the name attribute from a drop down - is there any point in having it. Could small instruction boxes be added to more complex forms to help guide the user. When data is returned, more clarity as to what the data is would help - perhaps with titles so that the user is re-assured that they have selected the correct data. - when printed you would need a title at the top rather than printing the whole form *

Signed: Mr. Hewitt

Date: 25/02/16

and then inferring from the menus what the data is showing. Graph Axis are not labelled and hard to read.

User feedback from Mr Nicoll, a primary user and secondary user of the system.

User Feedback Form

Please rate the following aspects of the fully implemented system out of five:

	5: Unusable	4: Poor	3: Neutral	2: Good	1: Excellent
The User Interface (GUI)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Adding and editing Records within the system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Comparing Achieved and Predicted Grades	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Importing Student data to the system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Exporting Predicted grades from the system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Filtering Records within the system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Additional feedback on features such as database records structure, ease of use of the system, user privileges and data security etc.

Good distinction between user and admin.
 Appropriate functionality for each user group. Consistent GUI for all users.
 Possibly give the user the ability to select a teacher from a drop down list and display the teacher name at the top of the page with class details. A further development could be to compare achieved against predicted.

Signed: RPNicoll

Date: 25/2/2016

User feedback from Mr Hardy, a primary user of the system.

User Feedback Form

Please rate the following aspects of the fully implemented system out of five:

	5: Unusable	4: Poor	3: Neutral	2: Good	1: Excellent
The User Interface (GUI)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Adding and editing Records within the system	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Comparing Achieved and Predicted Grades	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Importing Student data to the system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Exporting Predicted grades from the system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Filtering Records within the system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Additional feedback on features such as database records structure, ease of use of the system, user privileges and data security etc.

Some small inconsistencies in user experience which can be marginally confusing.

Strong filtering options for all datagrid views

Grade export very useful for integration.

Comparing Achieved and predicted grades need work.

Signed: A. Hardy

Date: 25/02/2016

User feedback from Mr Harrison, a potential primary and/or secondary user of the system.

User Feedback Form

Please rate the following aspects of the fully implemented system out of five:

The User Interface (GUI)

	5: Unusable	4: Poor	3: Neutral	2: Good	1: Excellent
The User Interface (GUI)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Adding and editing Records within the system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Comparing Achieved and Predicted Grades	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Importing Student data to the system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Exporting Predicted grades from the system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Filtering Records within the system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Adding and editing Records within the system

Comparing Achieved and Predicted Grades

Importing Student data to the system

Exporting Predicted grades from the system

Filtering Records within the system

Additional feedback on features such as database records structure, ease of use of the system, user privileges and data security etc.

ERROR MESSAGE - WINDOW CAPTION NOT IN EXISTENCE.
 ↳ CURRENTLY SAYS '0'.

- WOULD BE USEFUL TO PROMPT USER TO SELECT ACADEMIC YEAR PRE - SELECTING FUNCTION (MOVE SELECT BOX HIGHER?)
- Data Validation - some inconsistency issues.
 ↳ Think about how you could add validation checks in.

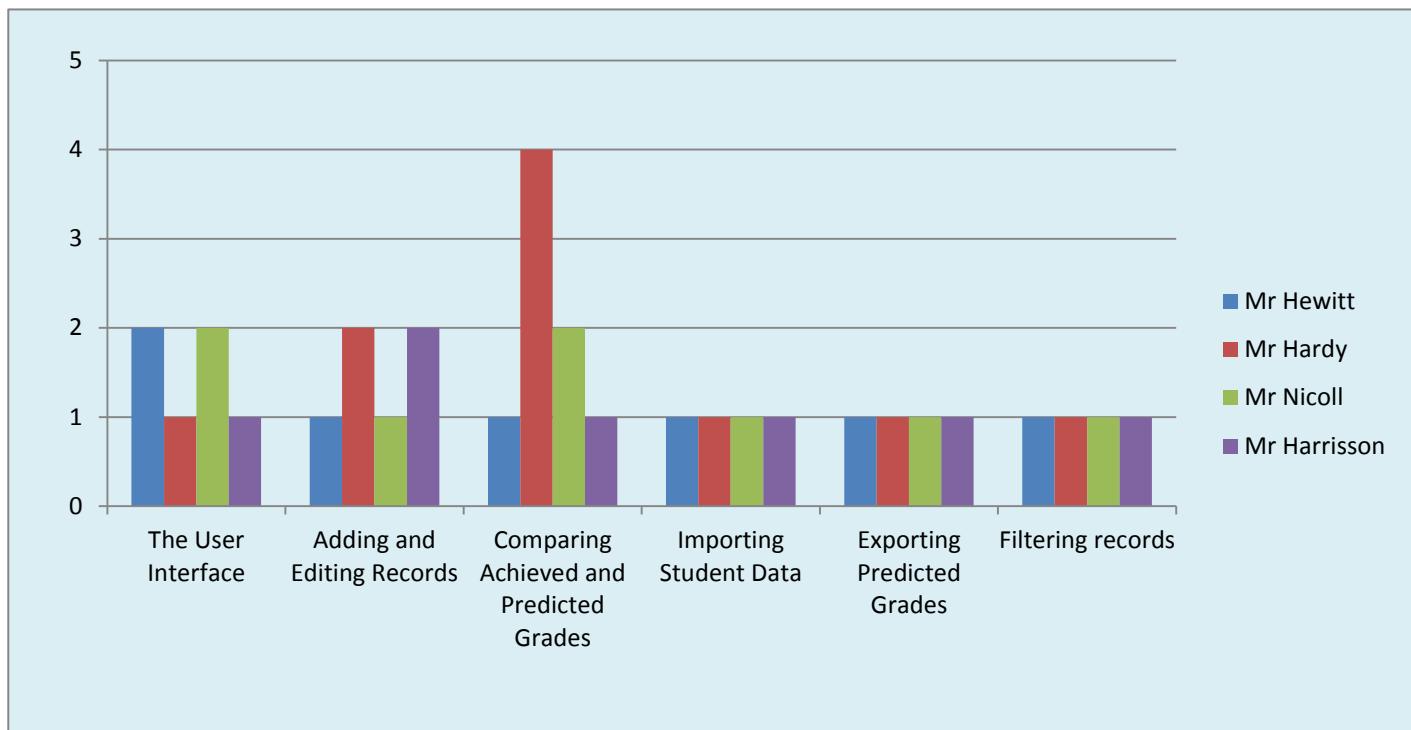
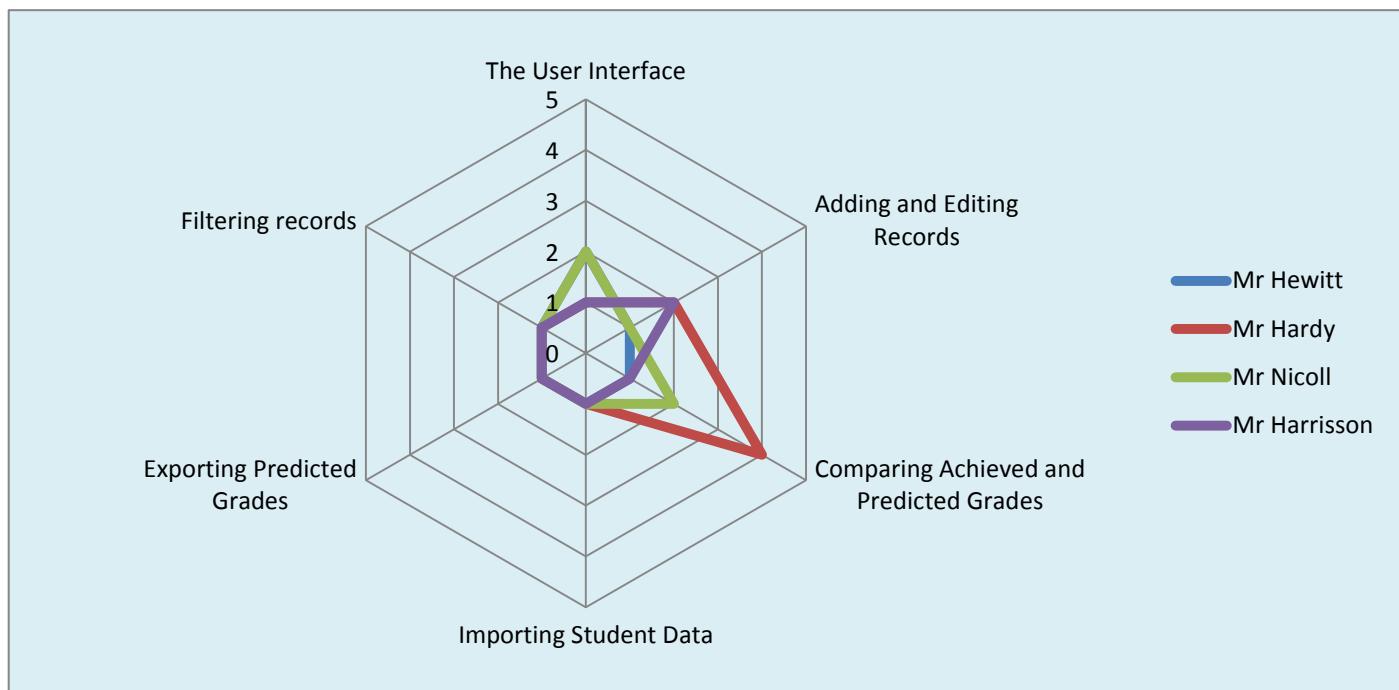
④ HOWEVER I MIGHT WANT TO COMPARE PREDICTED AGAINST ACHIEVED.

Signed: RL

Date: 22 feb . 16 .

Analysis of User Feedback

I designed a user feedback form which I gave to users of my system to gather end-user feedback of my implemented system. Along with the form I gave each user the installation files and user guide for them to test the system. I instructed them to rate my user interface, my data management, achieved and predicted grade features and system import and export capabilities out of five with one being the highest. I also put an additional space on the user feedback form for my general comments about the system. I have summarised their comments and potential further development ideas on the following page:



User Feedback Analysis			
Mr Hewitt	Mr Nicoll	Mr Hardy	Mr Harrison
<ul style="list-style-type: none"> Forms are well presented Message boxes are in a standard format and could benefit from being in the same format as main forms Some buttons can be clicked on multiple times and will not do anything unless other options are selected When filtering staff a user can only select the attribute from a drop down filter Could small instruction boxes be added to forms to help guide users? More clarity as to what data is when it is selected- perhaps add titles to say what data has been selected Graph axis are not labelled which makes graphs hard to read 	<ul style="list-style-type: none"> Good Distinction between user and admin Consistent GUI has been applied for all users Possibly allow the user to select a teacher from a combo box selection and then display all class details for that teacher 	<ul style="list-style-type: none"> Strong Filtering Options Grade export useful for further integration Comparing Achieved and Predicted Grades needs work. There are some inconsistencies in the user experience which could be marginally compromising. 	<ul style="list-style-type: none"> Error message windows captions don't exist Would be useful to prompt a user to select Academic Year when logging into system Move select academic year combo box higher I would want to be able to compare Achieved grades against Predicted Grades

Changes to consider

Error Message Windows's captions don't exist: In some cases within my system error messages and information messages don't have windows captions but some validation and information messages do. The one's where a window caption has not been specified default to a caption of "0". This is a problem which needs to be addressed in the future because at present there is an inconsistent use of windows captions.

Academic Year combo box: Mr Harrison suggested that when the user log's into the system they should be forced to select an academic year, however I don't agree with this change because not all forms in the system require an academic year to be selected. When an academic year is selected my system already displays an information message box requesting that the user complete an academic year selection. One thing I would do with the Academic Year combo box though is move it to the top of the main menu in a more appropriate position.

Comparing Achieved Grades against Predicted Grades: At present when a series is added to the comparison selection I restrict the user from using the grade type combo box selection, this means that a user cannot compare achieved grades against predicted grades. They can however compare achieved grades against achieved grades and predicted grades against predicted grades. I believe this was something I overlooked when I implemented my system because realistically a user may want to compare achieved grades against forecast grades, therefore I would un-restrict the use of the grade type combo boxes once a series has been added to allow for this in the future.

Possibly allow the user to select teacher from a comb-box selection and then display class details for that teacher: Currently an admin or user of my system can already search for a teacher via the teacher classes form and then select one of their classes. However I could make this feature more clear in the future by adding a dynamic title that says something along the lines of "Currently displaying Classes forYear ..." I could also add dynamic titles in other places across my forms to help specify what data grids represent, although a user could infer this by their filter selections.

Format error messages: This is something I would definitely implement in the future. I would format all my error messages and information messages in a similar way to how I have formatted my forms using the same backgrounds, button styles etc. This would allow for greater continuity across my system.

Graph axis labels: I believe that adding graph axis labels was a feature I overlooked during the implantation of my solution and I realize that by adding them it would allow users to understand my graph's easier. I would implement this in the future.

Small instruction boxes to aid users: This is a potential idea that could be implemented in accordance with my User documentation for the system. In order to solve this I could add an information icon to the top right corner of each form and then for each form display a set of instructions on how to use the current form.

Some buttons can be clicked multiple times with nothing happening until filters are updated: I believe that this is only a very minor problem with my system but I could solve this by displaying a validation message when the user clicks a button using the same previous filters.

Possible system extensions

Presently my system stores attendance data for all students but other than viewing a student's attendance record a user cannot use any system features to analyse a student's attendance record. Therefore one possible extension I could add to my system solution could be to add functionality which allows an admin or a teacher user to compare attendance data. The most interesting use of the attendance data stored within the system could be made by comparing a student's attendance record against their achieved grades. In this way I think user's may be able to make conclusions about a student's achieved grades based on their attendance record because I believe there may be a correlation between them.

Another system extension that I could consider is in regard to the printing of data and producing hard-copies from the system. Currently my system has print icons on forms which contain data grids or graphs which a user may want to produce a hard-copy of. However currently my print feature prints the current form display which means that some records may be missed from the data grid filter selection and that there are unnecessary features printed alongside the data grids or graphs such as form button objects. A potential solution for this could be to use the much more advanced print and formatting features of Microsoft Excel to produce hard copies of useful data and graphs. I could do this in a similar way to how I currently export predicted grades from the system as seen on my predicted grades form. For example when a user clicks the print icon, instead of printing the current display of the form I could export the contents of the data grid as a CSV file which is Microsoft Excel compatible. For graphs I would have to find some method of exporting them in a compatible Microsoft Excel format. I could then force the program to open the newly created CSV file; from here a user would have more functionality of data.

A further possible extension to my system could be to allow the system to also store Achieved A2 grades because at present I only store achieved GCSE and KS2 grades and then generate predictions. By storing achieved predicted grades this could allow for a user of the system to compare a student's final achieved predicted grades against the grades they were predicted based on their achieved GCSE grades. From here I could then add further functionality such as indicators which show whether a student has or has not made progress based on their original predictions. Furthermore I could use these progress indicators to show which subjects, departments or classes have the best or worst overall progress indicators based on their actual A2 grades performance against their predicted A2 performance.

Evaluation

Now that my project has been completed and implemented fully I will evaluate my performance in different areas of the system implementation.

Analysis- For my analysis I effectively managed to hold an interview with one of my system end-users and asked appropriate questions to find out information about both the current system alongside thoughts for the proposed system. After the interview I made a condensed summary of all the information I had found, allowing me to draw upon this information during implementation of the solution. Following my interview, I requested that I could have a look at the existing system in use. This allowed me to gain an insight into some of the features that the system should include and also how I could perhaps implement these features better. Another benefit from seeing the current system in use was that it allowed me to identify the usability of the solution in terms of its design. Another useful point of analysis was the entity relationship diagram of the current system because this helped me to finally implement and design my completed entity relationships within my database storage solution. Following all the analysis I had made of the current system and my exploration of the problem I managed to draw out some essential objectives for my proposed system that I intended to achieve. These objectives would later come in use when I implemented the solution, and since they were all SMART targets I was able to achieve them all within my solution time period. Overall, I feel as though I had ample evidence from my analysis which allowed me to implement a system that not only utilises the same features as the current system but also improves upon it with many more features.

Design- The design of my system was very important; which is why I put a lot of effort in to planning how I would implement the new system, discussing both the pros and cons of different implementation strategies. I managed to design my database relations in third normal form which was very important in order to make sure that all data stored within the system would be atomic. I also thought that I effectively planned my user interface prototypes, going into as much detail where possible. Since I had done this I was able to refer to my GUI interface prototypes closely to develop my forms in a way that made sense and had continuance. Another area I thought I had done well was designing my test plan; initially I was not too sure how I could test the navigation between forms within my system. Regardless, I designed a formatted table that would allow me to show the results of navigation testing in an understandable way. Lastly, I found that planning my algorithms and identifying algorithms I would have to use was useful because when it came to implementation of the system I referred closely to the pseudo codes I had written.

Implementation- The implementation of the system was a very extensive task which took me over 3 months to complete, consisting of over 170 pages of code (including annotation). It was not only the implementation of algorithms that I found difficult but I had to overcome certain form formatting problems. My finished solution though remained consistent throughout which is an essential feature of software. Furthermore, there was lots of extra validation which I implemented during the development of the system, some of which was implemented following an interview with one of my primary client's during development. This validation helps to make the system more practical and more importantly prevents data inconsistency across the entire system. Another thing that I implemented well was modular code within the program which would make maintaining the system much easier for a systems administrator.

Testing- Please see the testing evaluation discussed in the testing section.

User Guide- My user-guide was created in a consistent format with plenty of actual screenshots of the system so that a user could refer to examples of certain procedures in the system. I believe that my instructions provided were in depth and detailed enough for the user to follow effectively. I also included some examples of error messages that a user may receive so that they can recognize that they are a part of the system along with instructions on how to respond to errors.

Time Management- In order to successfully manage the implementation of such a large project it was very important that I managed my time effectively and set completion objectives for different sections of the coursework project. Before even starting the project I knew that the implementation along with the testing of the system would be two considerably large sections of the project. I began the Project at the beginning of September with a completion date in mind for April 29th. Accordingly I gave myself 1 and a half months to complete both of the analysis and design sections of my project. This then brought me to mid-November, from here I was not too sure how long it would actually take to implement the solution because at this point I could not be aware of what problems I may face during system implementation. Accordingly I set myself an objective to have the programmed solution and the testing completed by the middle of February, approximately 3 months. By the end of January though I had managed to complete my programmed solution and testing, but I needed to document my technical solution at this point. From here I still needed to complete the appraisal sections, user guide and implementation documentation. Appropriately I set myself two more months to have the coursework finished, but I managed to finish within just over a month because of the amount of hours I was spending per week on the coursework.

COMP4: YEAR 11 DYNAMIC DATA – USER MANUAL

Leon Singleton
Candidate No. 9178
Centre No. 23216
The Ecclesbourne School

Contents

INTRODUCTION TO THE SYSTEM.....	4
THE BENEFITS OF "Y11 DYNAMIC DATA" OVER OTHER SIMILAR SYSTEMS.....	4
INSTALLATION GUIDE	5
SETTING UP THE DATABASE	6

Instructions for Admin and Non-Admin user's

LOGGING INTO THE SYSTEM	9
USING THE MAIN MENU	10
SELECTING AN ACADEMIC YEAR.....	11
LOGGING OUT OF THE SYSTEM	11
PRINTING A FORM	12
NAVIGATING THE SYSTEM	12
VIEWING SUBJECT BASED STATISTICS	13
FILTERING A SUBJECT/SELECTING A SUBJECT	13
VIEWING THE RAW STATISTICAL DATA OF A SUBJECT	13
CLEARING THE CURRENT SELECTION.....	13
EXAMPLES OF ERROR MESSAGES	13
VIEWING PREDICTED GRADES.....	14
VIEWING A SINGLE STUDENTS PREDICTED GRADES	7
VIEWING MULTIPLE STUDENTS PREDICTED GRADES.....	14
CLEARING THE CURRENT SELECTION.....	14
EXPORTING PREDICTED GRADES FROM THE SYSTEM.....	14
COMPARING SUBJECTS STATISTICS.....	15
FILTERING A SUBJECT/SELECTING A SUBJECT	15
VIEWING THE RAW STATISTICAL DATA OF A SUBJECT	15
CLEARING THE CURRENT SELECTION.....	15
EXAMPLES OF ERROR MESSAGES	15
COMPARING DEPARTMENT STATISTICS	16
FILTERING A DEPARTMENT/SELECTING A DEPARTMENT.....	16
VIEWING THE RAW STATISTICAL DATA OF A DEPARTMENT.....	16
CLEARING THE CURRENT SELECTION.....	16
EXAMPLES OF ERROR MESSAGES	16
COMPARING CLASS STATISTICS	17
FILTERING A CLASS/SELECTING A CLASS	17
VIEWING THE RAW STATISTICAL DATA OF A CLASS.....	17
CLEARING THE CURRENT SELECTION.....	17
EXAMPLES OF ERROR MESSAGES	17
ADDING STUDENT'S TO CLASSES.....	18

EXAMPLES OF ERROR MESSAGES	18
REMOVING STUDENT'S FROM CLASSES	19
EDITING GRADES	19
EXAMPLES OF ERROR MESSAGES	19

Instructions for Non-Admin user's

VIEWING MY CLASSES.....	20
VIEWING MY CLASSES.....	20
ADDING A STUDENT TO ONE OF MY CLASSES.....	20
VIEWING ALL THE STUDENTS WITHIN ONE OF MY CLASSES	20
REMOVING A STUDENT FROM ONE OF MY CLASSES	20
CLEARING THE CURRENT SELECTION.....	20
VIEWING MY CLASSES GRADES	21
VIEWING MY CLASSES GRADES	21
VIEWING THE GRADES OF STUDENTS IN ONE OF MY CLASSES	21
EDITING THE GRADE OF ONE OF THE STUDENT'S IN MY CLASS.....	21
VIEW ALL OF A STUDENT'S GRADES.....	21
CLEARING THE CURRENT SELECTION.....	21

Instructions for Admin user's

VIEWING STUDENT'S DETAILS.....	22
FILTERING A SUBJECT/SELECTING A SUBJECT.....	22
EDITING A STUDENT'S DETAILS	22
EXAMPLES OF ERROR MESSAGES	22
ADDING NEW STUDENTS	23
ADDING A SINGLE STUDENT	23
ADDING MULTIPLE STUDENTS.....	23
ADDING MULTIPLE STUDENTS (CONTINUED).....	24
EXAMPLES OF ERROR MESSAGES (ADDING STUDENTS).....	24
VIEWING TEACHERS WITHIN THE SYSTEM	25
VIEWING TEACHER'S WITHIN THE SYSTEM	25
EDITING A TEACHER'S DETAILS.....	25
ADDING A NEW TEACHER	25
STUDENT'S CLASSES.....	26
VIEWING A STUDENT'S CLASSES	26
ADDING A STUDENT TO A CLASS.....	26
VIEWING ALL STUDENTS WITHIN A CLASS	27
REMOVING A STUDENT FROM A CLASS	27
EDITING AND ADDING CLASSES	28
VIEWING CLASSES WITHIN THE SYSTEM	28
EDITING A CLASS WITHIN THE SYSTEM	28
EXAMPLES OF ERROR MESSAGES	28

ADDING A NEW CLASS TO THE SYSTEM	29
EXAMPLES OF ERROR MESSAGES	29
FILTERING GRADES WITHIN THE SYSTEM	30
FILTERING A SELECTION OF GRADES	30
EDITING A GRADE	30
CLEARING THE CURRENT SELECTION.....	30
VIEW ALL GRADES FOR A SELECTED STUDENT	31
SELECTING A STUDENT.....	31
EDITING A GRADE	31
ADDING A NEW GRADE FOR A SELECTED STUDENT	32
EXAMPLES OF ERROR MESSAGES	32
ADVANCED ADMIN FEATURES	33
SELECTING AN ADMIN FUNCTION	33
DELETING AN ADMIN RECORD	33
ADDING A NEW ADMIN RECORD	33
FREQUENTLY ASKED QUESTIONS (FAQ)	34

Introduction to the system

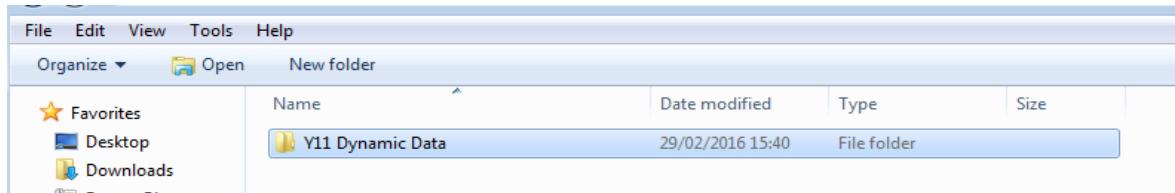
The year 11 Dynamic Data system allows tracking and analysis for year 11 students after the completion of an academic year. The system allows both admin use and normal teacher use. The system allows for data about student's to be stored including details of all the classes the students are in. Furthermore the system is able to store a student's achieved GCSE grade; a user can add or edit any of the above information. The system also allows for users to generate predicted grade data based on a student's GCSE performance. Extensive analysis can also be carried out by the system; this analysis is displayed graphically and can be used to compare achieved grade performance against achieved grade performance or predicted grade performance against predicted grade performance. When comparing data the raw statistical values of data can also be viewed within formatted table views. The system also incorporates a number of admin features which allow for extensive functionality of the system.

The benefits of “Y11 Dynamic Data” over other similar systems

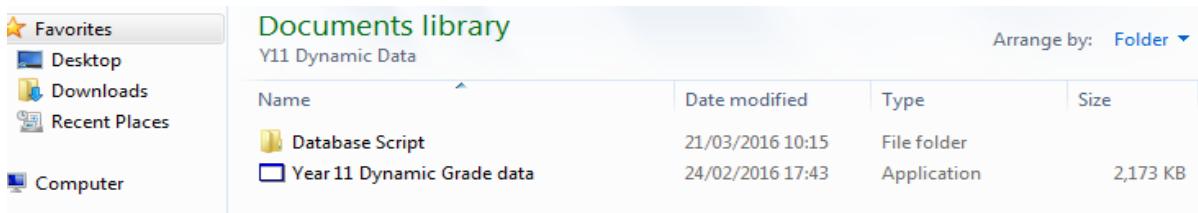
Since “Y11 Dynamic Data” is software packaged solution it allows for bespoke functionality that a client of the system can request. Moreover, the system is extremely secure because in order to use the system a machine with the school must be used because it must have a network connection to the SQL server. Furthermore, the system is password-protected and the database server is also password-protected. “Y11 Dynamic Data” is also unique in its grade predictions model, allowing for more accurate and realistic student grade predictions. Most systems take an average mean point score of a student's GCSE grades and then apply a prediction using a relative difficulty based on the A2 subject. However, “Y11 Dynamics Data” uses a weighting based on the relative difficulty of each GCSE grade and then generates a mean point score. From there it does a similar calculation in order to generate a predicted using the mean point score and applying A2 subject weightings.

Installation Guide

In order to first locate the file you must access it via the Staff shared network area and locate the folder named “Y11 Dynamic Data” and then double click the folder to view its contents.



Once you have opened the folder you can either run the program by double clicking it within this folder or you can copy and paste the executable to your user area for future use.



Alternatively, a copy of the program can be requested from ICT services, this will be installed on a USB stick. Once you have your USB stick you must plug it into your machine. From there once opening the USB's file browser you should have the same view as in the picture above. Again, you can either run the program by double clicking it within the file browser or you can copy and paste the executable to your user area for future use.

Setting up the Database (advanced)

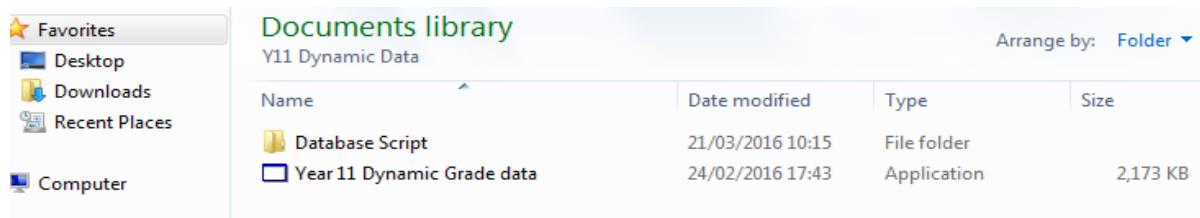
If you do not currently have a database server set up, then follow the steps in this section to set up an example database with the correct table relations and fields that will be compatible with the software. This is very important because if the names of individual fields do not match the ones used in the system then there will be a series of runtime errors. It is therefore of imperative importance that you use the sample database that I will provide and give instructions on how to set up. You will also need the following system requirements to set up your database:

System requirements

- Microsoft SQL Server Management Studio 2008 and newer
- Windows Based Operating System, can include: Windows 10, Windows 8, Windows 8.1, Windows 7, Windows Server 2012, Windows server 2008 R2
- 1.6ghz or faster processor
- 1gb of RAM (1.5gb if running on a virtual machine)
- 10GB of available hard disk space
- 5400 RPM hard disk space
- DirectX 9-capable video card that runs at 1024 x 768 or higher display resolution

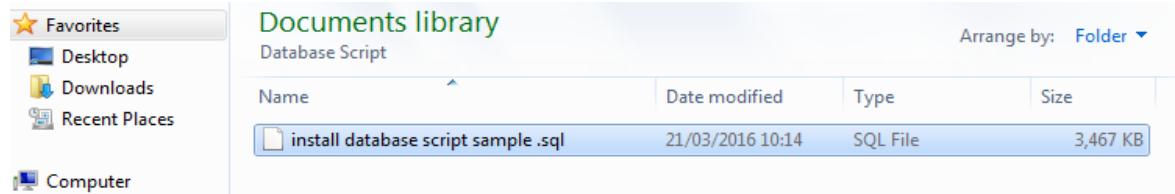
Finding the Install Script

First of all you must access the Y11 Dynamic Data folder which is located on the staff shared drive. From here you need to go inside of the “Database Script” folder which will contain a sample database example.



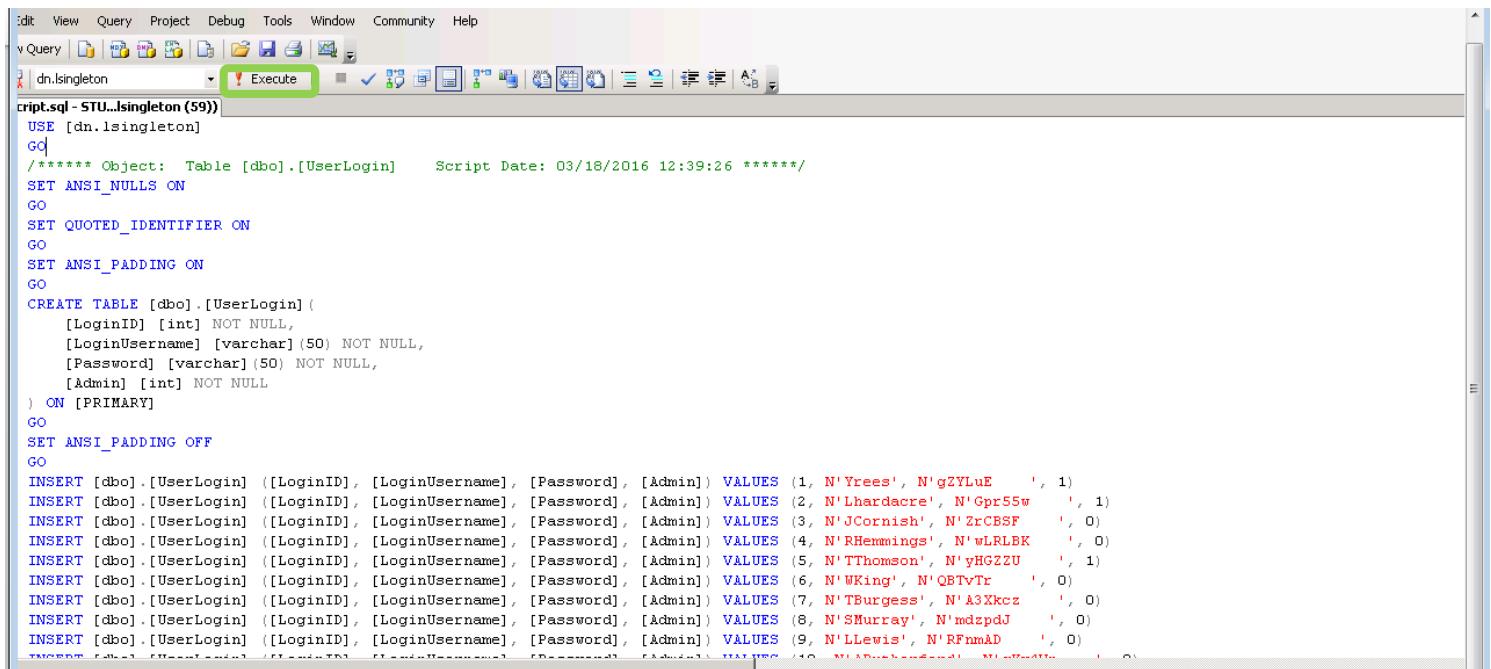
Then you need to double click on the “install database script sample” script file, which will re-direct you to an instance of Microsoft SQL server Management Studio.

Note: You must have installed Microsoft SQL server Management Studio before attempting to open the script



Once you have double clicked on the script file and you have successfully installed Microsoft SQL Server Management Studio on your system then you should have a screen that looks something similar like the screen shown in the example below. You may have to log into the SQL server first to see the screen below, you also need to make sure that you have created a blank database within your Management Studio named “dn.lsingleton”, From here you need to click the execute button (highlighted below), the script will then install and you will have a sample database with 3 years of sample data included.

Note: Make sure you have created a blank database named “<>” and make sure your database is on the “<>” server. Make sure the account you have used to create the database has the username “<>” and the password “<>”.



```

Edit View Query Project Debug Tools Window Community Help
Query | dn.lsingleton | ! Execute | 
script.sql - STU...lsingleton (59)
USE [dn.lsingleton]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[UserLogin] (
    [LoginID] [int] NOT NULL,
    [LoginUsername] [varchar](50) NOT NULL,
    [Password] [varchar](50) NOT NULL,
    [Admin] [int] NOT NULL
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
INSERT [dbo].[UserLogin] ([LoginID], [LoginUsername], [Password], [Admin]) VALUES (1, N'Yrees', N'gZYLuE', 1)
INSERT [dbo].[UserLogin] ([LoginID], [LoginUsername], [Password], [Admin]) VALUES (2, N'Lhardacre', N'Gpr55w', 1)
INSERT [dbo].[UserLogin] ([LoginID], [LoginUsername], [Password], [Admin]) VALUES (3, N'JCornish', N'ZrCBSF', 0)
INSERT [dbo].[UserLogin] ([LoginID], [LoginUsername], [Password], [Admin]) VALUES (4, N'RHemmings', N'wLBLBK', 0)
INSERT [dbo].[UserLogin] ([LoginID], [LoginUsername], [Password], [Admin]) VALUES (5, N'TThomson', N'yHGZ2U', 1)
INSERT [dbo].[UserLogin] ([LoginID], [LoginUsername], [Password], [Admin]) VALUES (6, N'WKing', N'QBTvTc', 0)
INSERT [dbo].[UserLogin] ([LoginID], [LoginUsername], [Password], [Admin]) VALUES (7, N'TBurgess', N'A3XkcZ', 0)
INSERT [dbo].[UserLogin] ([LoginID], [LoginUsername], [Password], [Admin]) VALUES (8, N'SMurray', N'mdzpdJ', 0)
INSERT [dbo].[UserLogin] ([LoginID], [LoginUsername], [Password], [Admin]) VALUES (9, N'LLewis', N'RFnmAD', 0)

```

Changing the database name or using a different server (advanced)

If you want to set up the database without using the standard server or database details then you can do this, but be aware that this will cause the system to not work and the program will say it cannot connect to a database. This is because the connection string within the program and the database will be incorrect. Therefore, you will need to locate the connectinostring variable property within the programmed solution and manually edit it to match the connection required to reach your server. There are two instances within the code where these need to be set, these are within the public module and the login form code. The two you need to change should like the ones below.

```

connectionstring = "Data Source=<>;Initial Catalog=<>;User ID=<>;Password=<>"

con.ConnectionString = "Data Source=<>;Initial Catalog=<>;User ID=<>;Password=<>"
```

For username and password details please contact a systems administrator, contact details found at the end of this user guide.

Database Table Creation Queries

Below are the table creation SQL queries that can be used to generate the correct database relations within Microsoft SQL Server Management Studio. These queries will create the tables and fields within the tables only, you will need to use the program to then insert records. In order to insert one of these queries you need to open a new query within Microsoft SQL Server Management Studio and then copy/paste these queries in separately and execute them for each table relation.

Note: Do not use these if you have used the sample database

```
CREATE TABLE [UserLogin](
```

```
    [LoginID] [int] NOT NULL,
```

```
    [LoginUsername] [varchar](50) NOT NULL,
```

```
    [Password] [varchar](50) NOT NULL,
```

```
    [Admin] [int] NOT NULL)
```

```
CREATE TABLE [Teaches](
```

```
    [TeachesID] [int] NOT NULL,
```

```
    [TeacherID] [int] NOT NULL,
```

```
    [ClassID] [int] NOT NULL)
```

```
CREATE TABLE [Teacher](
```

```
    [TeacherID] [int] NOT NULL,
```

```
    [LoginID] [int] NOT NULL,
```

```
    [Firstname] [varchar](50) NOT NULL,
```

```
    [Surname] [varchar](50) NOT NULL)
```

```
CREATE TABLE [Subject](
```

```
    [SubjectID] [int] NOT NULL,
```

```
    [DepartmentID] [int] NOT NULL,
```

```
    [Subject] [varchar](50) NOT NULL)
```

```
CREATE TABLE [StudentGroup](
```

```
    [StudentGroupID] [int] NOT NULL,
```

```
    [ClassID] [int] NOT NULL,
```

```
    [StudentID] [int] NOT NULL)
```

```
CREATE TABLE [Student](
```

```
    [StudentID] [int] NOT NULL,
```

```
    [Firstname] [varchar](50) NOT NULL,
```

```
    [Surname] [varchar](50) NOT NULL,
```

```
    [Gender] [varchar](1) NULL,
```

```
    [FSM] [varchar](1) NULL,
```

```
    [SNS] [varchar](1) NULL,
```

```
    [Ethnicity] [varchar](50) NULL,
```

```
    [Attendance] [float] NULL,
```

```
    [DateOfBirth] [date] NULL,
```

```
    [Form] [varchar](5) NULL,
```

```
    [Year] [int] NULL,
```

```
    [DatasetID] [int] NULL)
```

```
CREATE TABLE [Level](
```

```
    [LevelID] [varchar](5) NOT NULL,
```

```
    [Level] [varchar](5) NOT NULL)
```

```
CREATE TABLE [GradeType](
```

```
    [GradeTypeID] [int] NULL,
```

```
    [GradeType] [varchar](20) NULL)
```

```
CREATE TABLE [Grade](
```

```
    [GradeID] [int] NOT NULL,
```

```
    [StudentID] [int] NOT NULL,
```

```
    [SubjectID] [int] NOT NULL,
```

```
    [LevelID] [int] NOT NULL,
```

```
    [GradeTypeID] [int] NOT NULL,
```

```
    [Grade] [varchar](10) NOT NULL,
```

```
    [DatasetID] [int] NOT NULL)
```

```
CREATE TABLE Department(
```

```
    [DepartmentID] [int] NOT NULL,
```

```
    [Department] [varchar](50) NOT NULL)
```

```
CREATE TABLE [DatasetID](
```

```
    [DatasetID] [int] NULL,
```

```
    [Year] [varchar](5) NULL)
```

```
CREATE TABLE [Class](
```

```
    [ClassID] [int] NOT NULL,
```

```
    [SubjectID] [int] NOT NULL,
```

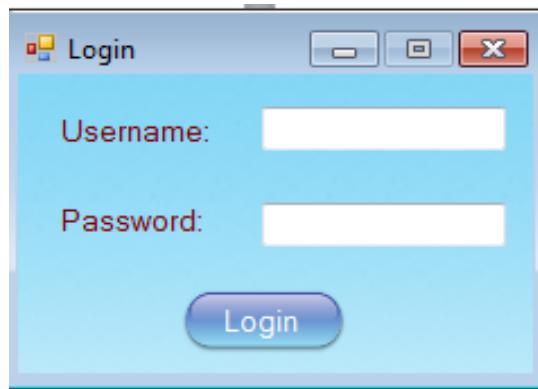
```
    [TeacherID] [int] NOT NULL,
```

```
    [ClassGroup] [varchar](17) NOT NULL,
```

```
    [DatasetID] [int] NOT NULL)
```

Logging into the System

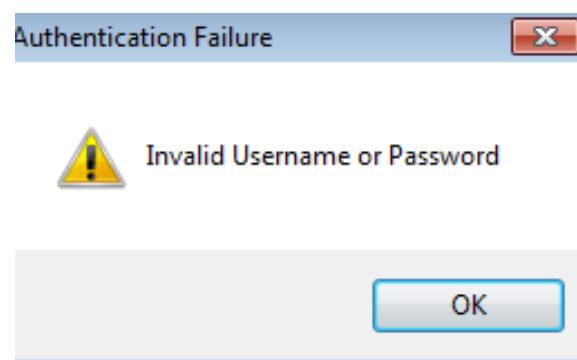
Once you have installed the program and have double clicked the program's executable you will be welcomed with a login form that looks like the picture opposite.



From here you are required to enter a valid set of login details into both the username and password's field. Once you have entered your username and password into the form's corresponding fields you can login into the system by clicking the login button. Admin privileges will be automatically applied if you are applicable. An example of logging into the form can be seen below.



However, if you attempt to login into the system using invalid login information then an error message will inform you that invalid login details have been entered and the text fields for both username and password will be reset to blank.



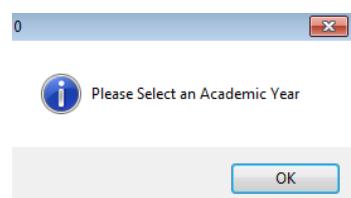
Using the Main Menu

In order to navigate the main menu, you can click on one of the boxes, for example if you click on the grade predicts button you will be directed to the predicted grades form. Where a form requires you to select an academic year you will be prompted to do so. An example of this error message can be seen further down below if you attempt to navigate to a form that requires an academic year to be selected.



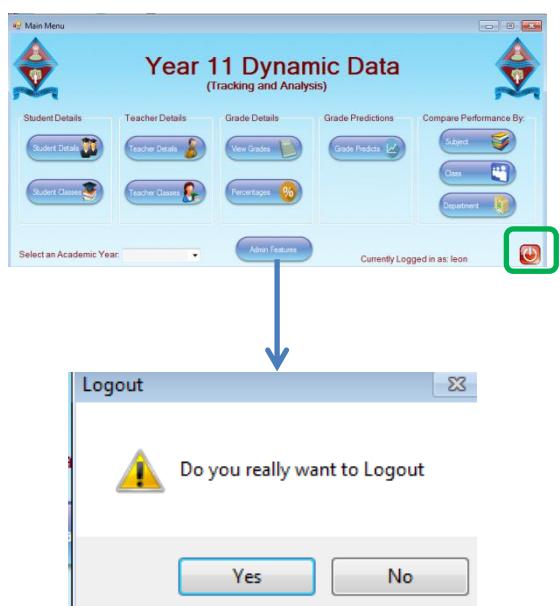
Selecting an Academic Year

If you receive the error message to the right then you have not selected an academic year. In order to select an academic year you must select one from the combo box selection list (highlighted above in green).



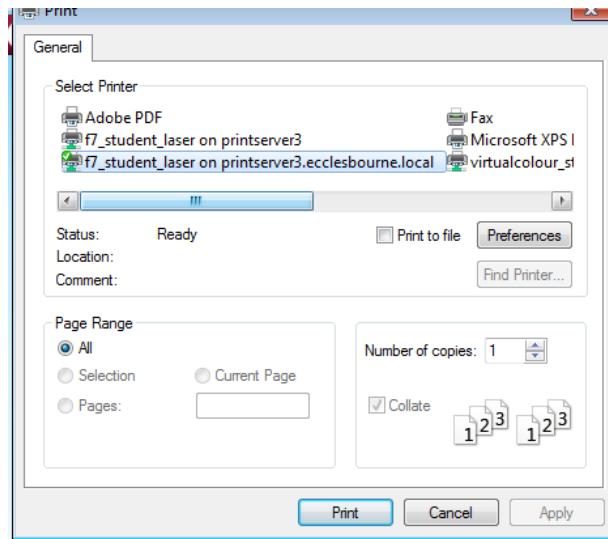
Logging out of the system

If you want to log out of the system then you can do so by clicking the icon in the bottom right corner of the main menu. I have highlighted this in the form opposite. Once you have clicked the logout icon you will be asked if you really want to log out. Clicking yes will log you out of the system and send navigate you to the login form and clicking no will do nothing.



Printing a form

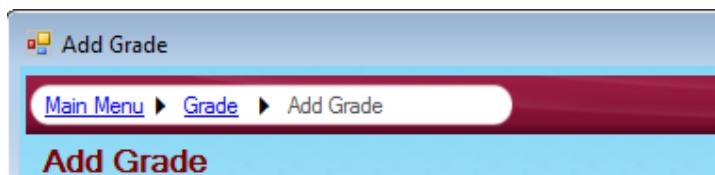
In order to print a form you must first of all click the print icon which is found in the top right corner of a printable form and looks like the icon opposite.



When you have clicked the print icon you will see a print dialog screen which looks like the one above. From here you can select the number of copies to print amongst other printing preferences. You must select the printer you want to print to before you click the print button. Note it will be your set as your default system printer. Once you are happy with your printing options you can print your form by clicking the print button. A printed copy of the form will be produced in landscape orientation.

Navigating the System

You will notice that in the top left hand corner of the screen there is a bread crumb trail which indicates the current position of the form you are on and the forms that you have had to go through to reach the form you are on. In order to navigate back to the main menu or a previous form you can click the link in the breadcrumb trail. Clickable links are blue.



In the above example a user could navigate back to the grade form by clicking the "grade" hyperlink or the main menu by clicking the "Main Menu" hyperlink.

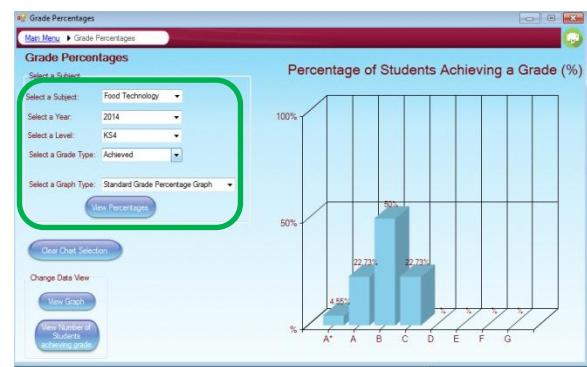
Viewing Subject Based Statistics

First of all you must select the percentages button from the main menu form which is the same as the button opposite.



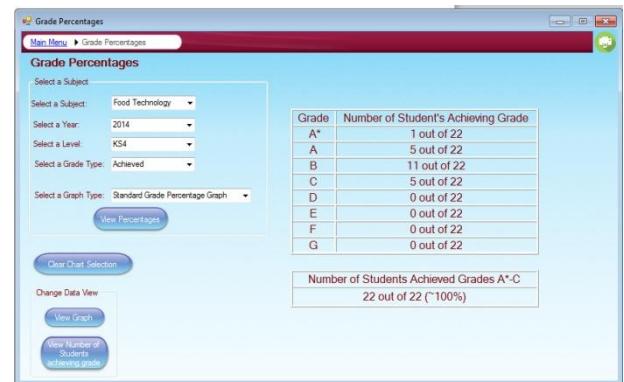
Filtering a subject/selecting a subject

In order to select a subject you must use the combo boxes within the group box which is titled “select a subject”. Then once you have selected a subject’s statistics which can either be achieved or predicted you need to select the graph type. You can select either a linear or cumulative graph from the graph format selection combo box. Finally you must click the “View percentages” button.



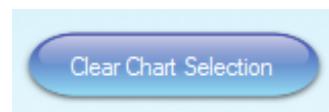
Viewing the raw statistical data of a subject

In order to view the raw statistical data of a subject you can change the data view by clicking the “view number of student’s achieving grade” button. Upon clicking this, the form will look something like the one opposite. In order to return to the graph view you can click the “view Graph” button.



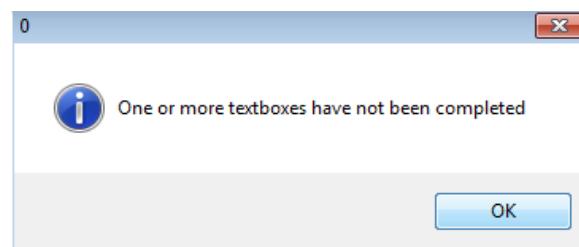
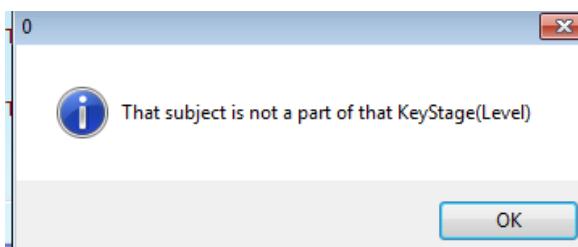
Clearing the current selection

In order to clear the current graph selection and the current filter selections you can click the “Clear chart selection” button.



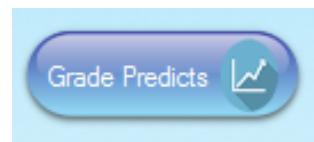
Examples of error messages

The following error messages are quite self-explanatory and you can ignore these messages by clicking the ok button and amending the correct input boxes.



Viewing Predicted Grades

First of all you must select the Grade Predicts button from the main menu form which is the same as the button opposite.



Viewing a single students predicted grades

First of all you must search a student from the student filter selection and click the search button you can do this using several filters. When you search for a student, a data grid will appear which will contain a list of students that match your filter selection. To select a student you must click one of the “view Grades” buttons, click the button that is on the same row as the student you wish to select.

Two screenshots illustrating the search process. The left screenshot shows the search interface with a green box highlighting the search input field where 'will' has been typed. The right screenshot shows the results grid for students named 'will', with a green box highlighting the 'View Grades' button for the first student in the list.

Once you have selected a student you will then be able to see that entire student's achieved grades. To see their predicted grades you must click one of the “predicted grades buttons on the data grid. A separate data grid will then appear alongside the current grid, containing all of that student's predicted A2 grades.

Three screenshots illustrating the expanded view of predicted grades. The left screenshot shows the main search interface with a green box around the 'View Predicted Grades' button for the first student in the results grid. The middle screenshot shows the results grid with a green box around the 'View Predicted Grades' button for the first student. The right screenshot shows a separate data grid listing all of the student's predicted A2 grades, with a green box around the 'Predicted A2 Grade' column header.

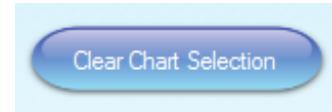
Viewing multiple students predicted grades

In order to view multiple students' worth of predicted grades you can do this by class, form or year. There are separate group boxes for each of these functions. For example if you wish to select a class like the example opposite you must fill in all the fields in the class group box and then click the search button within the class group box.

Firstname	Surname	Art	Biology	Business Studies	Chemistry	Computing	Drama	Economics	English Language	English Literature
Nathaniel	Finlay	B	C	B	C	C	B	B	B	B
Benoit	Hannah	A	C	B	C	C	B	B	B	B
Andrew	Emma	A	C	A	C	B	B	A	B	B
Matthew	Holly	C	D	C	D	D	C	C	D	D
Tom	Jess	B	D	B	D	C	C	B	C	C
Ben	Kate	B	C	B	C	C	B	B	C	B
George	Leanne	A	B	A	B	B	A	A	A	A
Ben	Maisie	A	C	B	C	C	B	B	B	B
Will	William	B	C	B	C	C	B	B	B	B
Alexander	Christopher	D	E	D	E	E	D	D	D	D
Eve	Peter	A	C	A	C	B	B	A	B	B
Millie	Natalie	B	D	B	D	C	C	B	C	C
Jasmine	Megan	A*	B	A	B	B	A	A	A	A
Debbie	Rebecca	B	C	B	C	C	B	B	B	B
Kate	Isabella	B	C	B	C	C	B	B	B	B
Leah	Lauren	B	C	B	C	C	B	B	B	B
Rebecca	Tristan	B	D	B	D	C	C	B	C	C
Isobel	Harriet	A	C	B	C	C	B	B	B	B
Emily	Joe	C	D	C	D	D	C	C	C	C
Lauren	Chloe	B	C	B	D	C	B	B	C	C
Abigail	Sophie	A	C	A	C	B	B	A	B	B

Clearing the current selection

In order to clear the current graph selection and the current filter selections you can click the "Clear chart selection button".



Exporting Predicted grades from the system

First of all you must click the "create spread sheet file" button which looks like the one opposite. This button will only become visible when you have displayed some predicted grades.



A save file dialog display will then appear which will allow you to save the newly created spread sheet (CSV) file in any location on your system. You should also name your file appropriately. Click the save button when you have found the destination you wish to save the file. To view the file in your default spread sheet program double click it in its file location. It should then look like the below Microsoft Excel example.

Firstname	Surname	Art	Biology	Business Studies	Chemistry	Computing	Drama	Economics	English Language	English Literature	ICT	Leisure
2 Nathaniel	Finlay	B	C	B	B	B	B	C	B	C	B	B
3 Benoit	Hannah	A	C	C	B	B	B	B	C	A	A	B
4 Andrew	Emma	A	C	B	B	A	B	B	B	A	A	B
5 Matthew	Holly	C	D	C	D	C	D	C	D	C	D	C
6 Tom	Jess	B	D	D	C	C	B	C	C	B	C	C
7 Ben	George	B	C	C	B	B	B	B	C	B	B	B
8 Ben	Maisie	A	B	B	B	A	A	A	B	A	A	A
9 Will	William	B	C	C	B	B	B	B	C	A	A	B
10 Alexander	Christopher	D	E	E	D	D	D	E	D	E	D	D
11 Eve	Peter	A	C	B	B	A	B	B	B	A	B	B
12 Millie	Natalie	B	D	D	C	C	B	C	C	B	C	C
13 Jasmine	Megan	A*	B	A	B	A	A	A	B	A*	A*	A
14 Debbie	Rebecca	B	C	C	B	B	B	C	B	B	C	B
15 Katie	Isabella	B	C	C	B	B	B	C	C	B	C	B
16 Leah	Lauren	B	C	C	B	B	B	C	C	B	C	B
17 Rebecca	Tristan	B	D	C	B	C	B	B	C	B	C	C
18 Isobel	Harriet	A	B	C	B	B	B	C	C	B	C	B
19 Emily	Joe	C	D	D	C	C	C	D	D	C	D	C
20 Lauren	Chloe	B	C	B	C	B	C	C	B	B	C	B
21 Abigail	Sophie	A	C	A	B	A	B	B	A	A	B	B

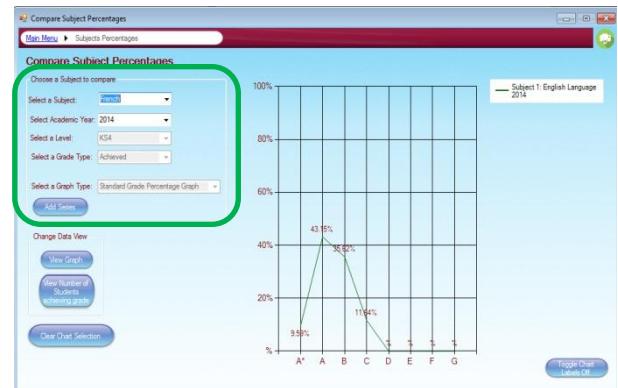
Comparing Subjects Statistics

First of all you must select the Subject button from the main menu form which is the same as the button opposite.



Filtering a subject/selecting a subject

In order to select a subject you must use the combo boxes within the group box which is titled “choose a subject to compare” (highlighted in green opposite). Then once you have selected a subject’s statistics which can either be achieved or predicted you need to select the graph type. You can select either a linear or cumulative graph from the graph format selection combo box. Finally you must click the “Display graph button” button. Once you have added a series you can add another series, up to a total of 4 series in the same way.



Viewing the raw statistical data of a subject

In order to view the raw statistical data of a subject you can change the data view by clicking the “view number of student’s achieving grade” button. Upon clicking this, the form will look something like the one opposite. In order to return to the graph view you can click the “view Graph” button.

Subject	Grade	Number of Student's Predicted Grade
Subject 1: Food Technology A2 Predicts 2015	A*	0 out of 209
	A	57 out of 209
	B	92 out of 209
	C	67 out of 209
	D	2 out of 209
	E	1 out of 209
	U	0 out of 209
	A*-C	206 out of 209 (~98%)
Subject 2: Geography A2 Predicts 2015	A*	0 out of 209
	A	69 out of 209
	B	93 out of 209
	C	41 out of 209
	D	2 out of 209
	E	0 out of 209
	U	0 out of 209
	A*-C	166 out of 209 (~79%)
Subject 3: History A2 Predicts 2015	A*	0 out of 209
	A	0 out of 209
	B	67 out of 209
	C	99 out of 209
	D	41 out of 209
	E	2 out of 209
	U	0 out of 209
	A*-C	166 out of 209 (~79%)
Subject 4: Maths A2 Predicts 2015	A*	0 out of 209
	A	0 out of 209
	B	57 out of 209
	C	97 out of 209
	D	53 out of 209
	E	2 out of 209
	U	0 out of 209
	A*-C	154 out of 209 (~73%)

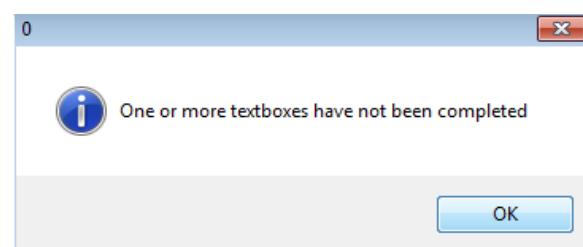
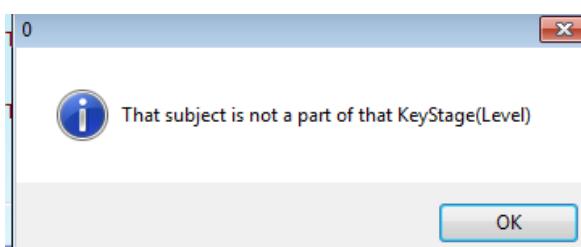
Clearing the current selection

In order to clear the current graph selection and the current filter selections you can click the “Clear chart selection” button.



Examples of error messages

The following error messages are quite self-explanatory and you can ignore these messages by clicking the ok button and amending the correct input boxes.



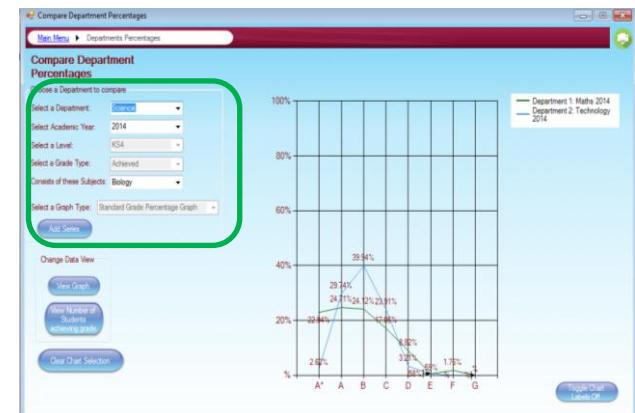
Comparing department Statistics

First of all you must select the Department button from the main menu form which is the same as the button opposite.



Filtering a Department/selecting a Department

In order to select a department you must use the combo boxes within the group box which is titled “choose a department to compare” (highlighted in green opposite). Then once you have selected a department’s statistics which can either be achieved or predicted you need to select the graph type. You can select either a linear or cumulative graph from the graph format selection combo box. Finally you must click the “Display graph button” button. Once you have added a series you can add another series, up to a total of 4 series in the same way.



Viewing the raw statistical data of a Department

In order to view the raw statistical data of a department you can change the data view by clicking the “view number of student’s achieving grade” button. Upon clicking this, the form will look something like the one opposite. In order to return to the graph view you can click the “view Graph” button.

Department 1: Maths 2014		Department 2: Technology 2014	
Grade	Number of Students achieving Grade	Grade	Number of Students achieving Grade
A*	38 out of 170	A*	9 out of 343
A	42 out of 170	A	102 out of 343
B	41 out of 170	B	137 out of 343
C	29 out of 170	C	82 out of 343
D	15 out of 170	D	11 out of 343
E	1 out of 170	E	2 out of 343
F	3 out of 170	F	0 out of 343
G	0 out of 170	G	0 out of 343
A*-C	151 out of 170 (~88%)	A*-C	330 out of 343 (~96%)

Department 3: Science 2014	
Grade	Number of Students achieving Grade
A*	16 out of 372
A	128 out of 372
B	159 out of 372
C	58 out of 372
D	7 out of 372
E	4 out of 372
F	0 out of 372
G	0 out of 372
A*-C	361 out of 372 (~97%)

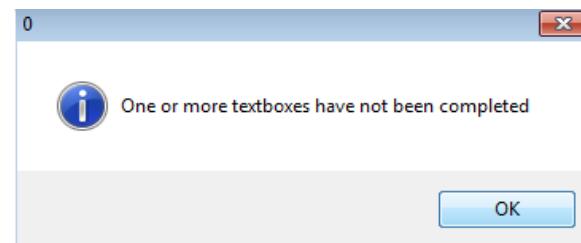
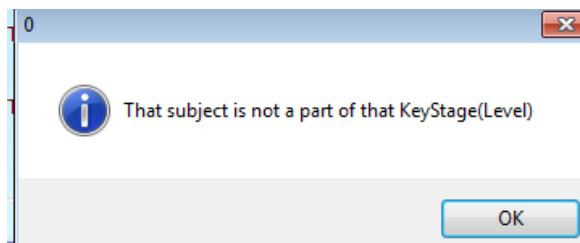
Clearing the current selection

In order to clear the current graph selection and the current filter selections you can click the “Clear chart selection” button.



Examples of error messages

The following error messages are quite self-explanatory and you can ignore these messages by clicking the ok button and amending the correct input boxes.



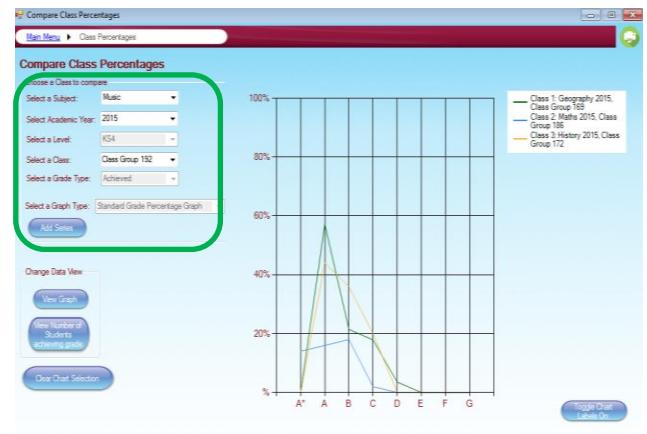
Comparing Class Statistics

First of all you must select the Class button from the main menu form which is the same as the button opposite.



Filtering a Class/selecting a Class

In order to select a Class you must use the combo boxes within the group box which is titled “choose a Class to compare” (highlighted in green opposite). Then once you have selected a Class’s statistics which can either be achieved or predicted you need to select the graph type. You can select either a linear or cumulative graph from the graph format selection combo box. Finally you must click the “Display graph button” button. Once you have added a series you can add another series, up to a total of 4 series in the same way.



Viewing the raw statistical data of a class

In order to view the raw statistical data of a class you can change the data view by clicking the “view number of student’s achieving grade” button. Upon clicking this, the form will look something like the one opposite. In order to return to the graph view you can click the “view Graph” button.

Class 1: Geography 2015, Class Group 169		Class 2: Maths 2015, Class Group 186		Class 3: History 2015, Class Group 172	
Grade	Number of Students achieving Grade	Grade	Number of Students achieving Grade	Grade	Number of Students achieving Grade
A*	0 out of 28	A*	7 out of 50	A*	0 out of 25
A	16 out of 28	A	8 out of 50	A	11 out of 25
B	6 out of 28	B	9 out of 50	B	9 out of 25
C	5 out of 28	C	1 out of 50	C	5 out of 25
D	1 out of 28	D	0 out of 50	D	0 out of 25
E	0 out of 28	E	0 out of 50	E	0 out of 25
F	0 out of 28	F	0 out of 50	F	0 out of 25
G	0 out of 28	G	0 out of 50	G	0 out of 25
A*-C	27 out of 28 (~96%)	A*-C	25 out of 50 (~50%)	A*-C	25 out of 25 (~100%)

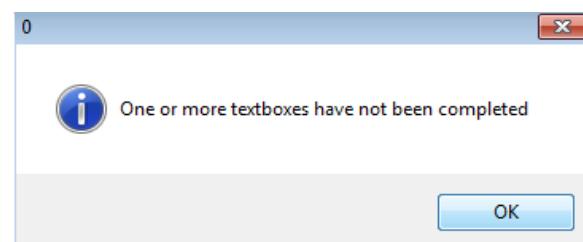
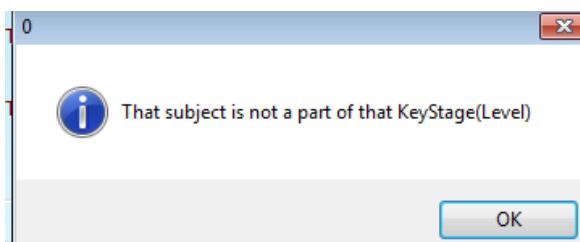
Clearing the current selection

In order to clear the current graph selection and the current filter selections you can click the “Clear chart selection” button.



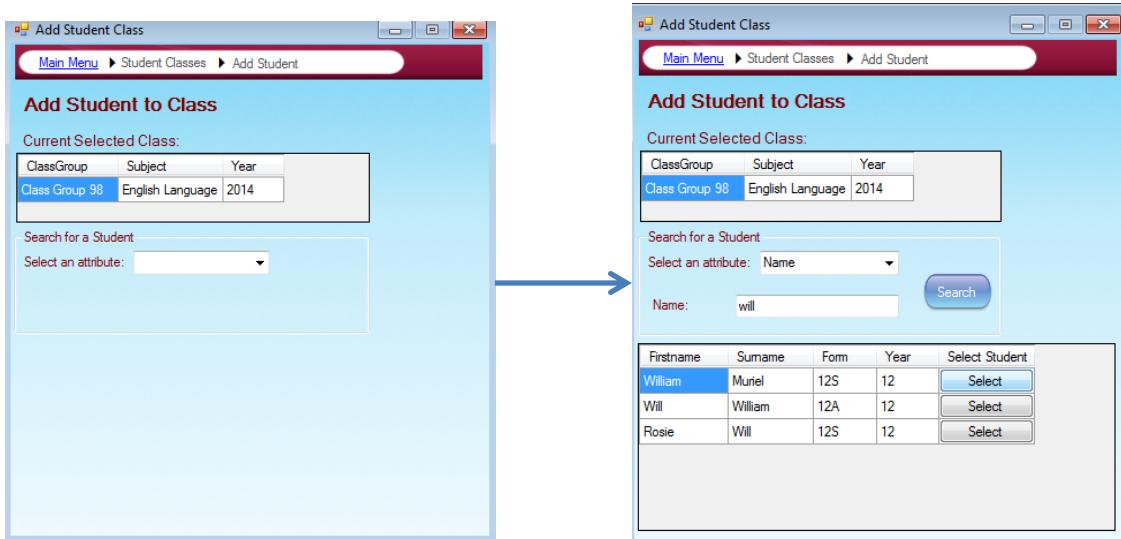
Examples of error messages

The following error messages are quite self-explanatory and you can ignore these messages by clicking the ok button and amending the correct input boxes.



Adding Student's to Classes

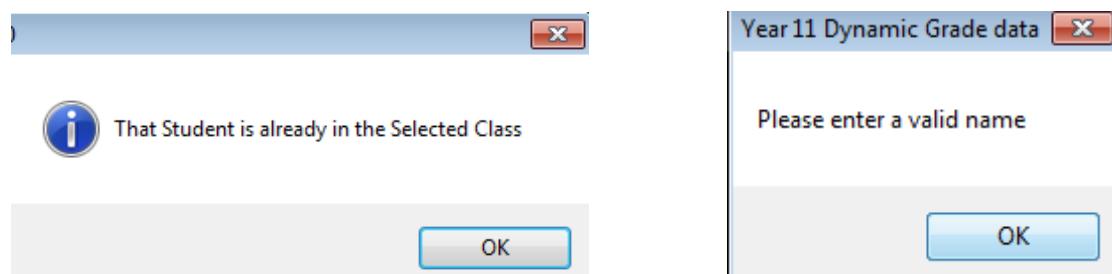
First of all you must look at the instructions on either the “my classes” form view or the Student’ Classes view and have selected a class that you wish to add a student to. You should have a similar pop-up window like the one below once you have done this. Then from here you need to search for a student by first selecting an attribute to filter by and then using the search box.



From here you must select the student that you want to add to the selected class by clicking one of the “select” buttons within the data grid. The “select” button you click will select the student on the same row as the button. Finally you can add the student to the selected class by clicking the “Add Selected Student” button.

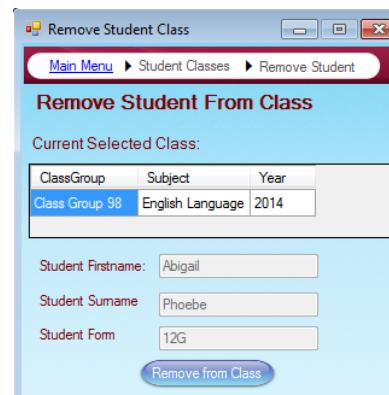
Examples of error messages

The following error messages are quite self-explanatory and you can ignore these messages by clicking the ok button and amending the correct input boxes.



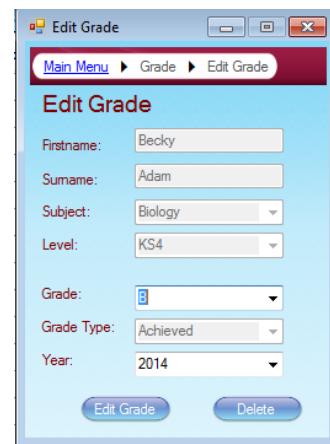
Removing student's from classes

First of all you must look at the instructions on either the “My Classes” form view or the Student’s classes form. Once you have the form opposite open and have successfully selected a student that you want to remove from a selected class you can remove them by clicking the “remove from class button”.



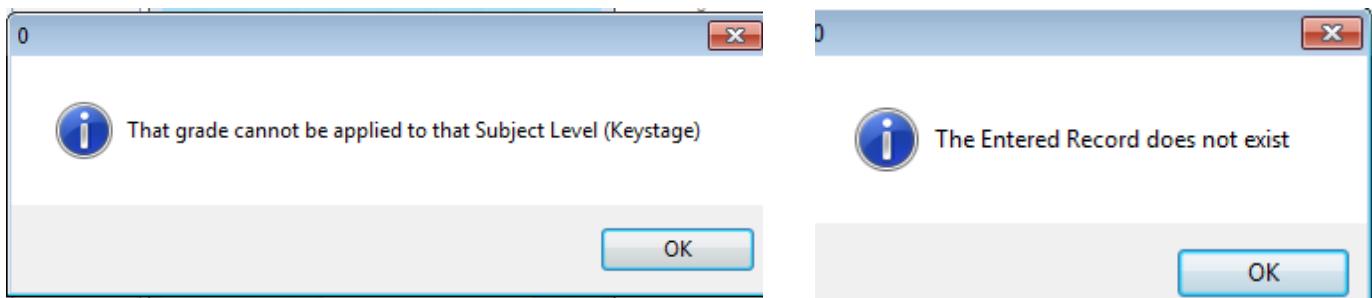
Editing Grades

First of all you must look at the instruction on either of the “My Class Grades” section, the “adding grades to students section” or the “filtering Grades” section. Once you have done this you will have a separate form open that looks like the one opposite. To delete a grade you can click the “delete” button and the grade you selected will be deleted from the system. To edit a grade you can either change the grade or the year of the grade achieved (for re-takes etc.), to confirm any changes you must click the “edit Grade” button.



Examples of error messages

The following error messages are quite self-explanatory and you can ignore these messages by clicking the ok button and amending the correct input boxes.



Viewing my Classes

First of all you must select the My Classes button from the main menu form which is the same as the button opposite.



Viewing my classes

When the form initially loads it should contain all the classes you have taught for a specific year, note that the classes within the data grid are the ones you taught for the academic year you selected on the main menu.

My classes						
ClassGroup	Subject	Department	Year	Firstname	Surname	View Current Class Add a Student to this Class
Class Group 98	English Language	English	2014	Andrea	Car	View Students Add Student
Class Group 101	English Literature	English	2014	Andrea	Car	View Students Add Student
Class Group 129	Biology	Science	2014	Andrea	Car	View Students Add Student

Adding a student to one of my classes

In order to add a student to one of your classes you must click the “add students” button which is on the same row as the class you want to add a student to. A new add student to class form will appear. From here please refer to the adding students to classes section of the user manual for further instructions.

Add Student Class

Main Menu	Student Classes	Add Student
Add Student to Class		
Current Selected Class:		
ClassGroup	Subject	Year
Class Group 101	English Literature	2014
Search for a Student		
Select an attribute:		

Viewing all the students within one of my classes

In order to view a list of all the students that are in one of your classes you must click the “view students” button which is on the same row as the class you want to view a list of students for. When you do this the data grid will update with a list of students that were in that class.

My classes						
Main Menu > My Classes						
My Classes						
Firstname	Surname	Form	ClassGroup	Subject	Department	Year
Angel	Phoebe	12D	Class Group 98	English Language	English	2014
Angel	Sophie	12A	Class Group 98	English Language	English	2014
Amy	Jane	12B	Class Group 98	English Language	English	2014
Any	Adam	12M	Class Group 98	English Language	English	2014
Bogdan	Arnold	12B	Class Group 98	English Language	English	2014
Caitlin	Hannah	12J	Class Group 98	English Language	English	2014
Charlotte	Joe	12P	Class Group 98	English Language	English	2014
Emily	Joe	12A	Class Group 98	English Language	English	2014
Emily	Oscar	12P	Class Group 98	English Language	English	2014
Francesca	Charlotte	12W	Class Group 98	English Language	English	2014
Gemma	Lizze	12B	Class Group 98	English Language	English	2014
George	Mabel	12A	Class Group 98	English Language	English	2014
George	Ben	12B	Class Group 98	English Language	English	2014
Hannah	Tom	12B	Class Group 98	English Language	English	2014
Holly	Ben	12M	Class Group 98	English Language	English	2014
Isabella	Amy	12J	Class Group 98	English Language	English	2014
Abigail	Ursula	12A	Class Group 98	English Language	English	2014

Removing a student from one of my classes

In order to remove a student you must click the “remove student” button which is on the same row as the student you want to remove from the class. A separate form will appear which will allow you to remove the selected student from the selected class. From here please refer to the “removing student’s from classes” section of the user manual.

Remove Student Class

Main Menu	Student Classes	Remove Student
Remove Student From Class		
Current Selected Class:		
ClassGroup	Subject	Year
Class Group 98	English Language	2014
Student Firstname:	Abigail	
Student Surname:	Sophie	
Student Form:	12A	
Remove from Class		

Clearing the current selection

In order to clear the current data grid selection you can click the “Clear selection” button.



Viewing my Classes Grades

First of all you must select the My Class Grades button from the main menu form which is the same as the button opposite.



Viewing my classes Grades

When the form initially loads it should contain all the classes you have taught for a specific year, note that the classes within the data grid are the ones you taught for the academic year you selected on the main menu.

My Classes								
ClassGroup	Subject	Department	Year	Firstname	Surname	View Current Students	Add a Student to this Class	
Class Group 98	English Language	English	2014	Andrea	Car	View Students	Add Student	
Class Group 101	English Literature	English	2014	Andrea	Car	View Students	Add Student	
Class Group 139	Biology	Science	2014	Andrea	Car	View Students	Add Student	

Viewing the Grades of students in one of my classes

In order to view a list of student's grades within one of your classes you must click the "view Grades" button on the row of the class you want to view student's grades. After clicking one of these buttons the data grid will update with a list of student's within the selected class and the grades they achieved for your class subject.

My Class Grades								
Firstname	Surname	Subject	Grade	Level	GradeType	Year	Edit Class Grade	View all Student's Grades
Abigail	Phoebe	English Language	A	KS4	Achieved	2014	Edit Grade	View Grades
Abigail	Sophie	English Language	A	KS4	Achieved	2014	Edit Grade	View Grades
Amelia-Jane	Georgina	English Language	A	KS4	Achieved	2014	Edit Grade	View Grades
Amy	Adam	English Language	A	KS4	Achieved	2014	Edit Grade	View Grades
Brianna	Arnold	English Language	B	KS4	Achieved	2014	Edit Grade	View Grades
Caitlin	Hannah	English Language	B	KS4	Achieved	2014	Edit Grade	View Grades
Charlotte	José	English Language	B	KS4	Achieved	2014	Edit Grade	View Grades
Emily	Emily	English Language	C	KS4	Achieved	2014	Edit Grade	View Grades
Emily	Grace	English Language	A	KS4	Achieved	2014	Edit Grade	View Grades
Francesca	Charlotte	English Language	B	KS4	Achieved	2014	Edit Grade	View Grades
Gemma	Liam	English Language	A	KS4	Achieved	2014	Edit Grade	View Grades
Georgia	Mabel	English Language	B	KS4	Achieved	2014	Edit Grade	View Grades
George	Ben	English Language	A	KS4	Achieved	2014	Edit Grade	View Grades
George	Tom	English Language	A*	KS4	Achieved	2014	Edit Grade	View Grades
Hannah	Ben	English Language	B	KS4	Achieved	2014	Edit Grade	View Grades
Hannah	Tom	English Language	B*	KS4	Achieved	2014	Edit Grade	View Grades
Iabella	Any	English Language	B	KS4	Achieved	2014	Edit Grade	View Grades
IABELLA	Hannah	English Language	A	KS4	Achieved	2014	Edit Grade	View Grades

Editing the grade of one of the student's in my class

To edit the grade of one of the student's within your class you must click the "Edit grade" button on the row of the student's grade you want to edit. Upon clicking one of these buttons a new form will appear which will allow you to edit the student's selected grade. For information on using the editing grade form refer to the editing grades section of the user manual.

Main Menu > Grade > Edit Grade

Edit Grade

Firstname:	Abigail
Surname:	Phoebe
Subject:	English Language
Level:	KS4
Grade:	A
Grade Type:	Achieved
Year:	2014
Edit Grade	
Delete	

View all of a student's grades

Click the view grades button on the same row as the student that you want to view all their achieve grades. This will allow you to see how a student within one of your classes may be performing elsewhere across the school.

My Class Grades								
Firstname	Surname	Subject	Grade	Level	GradeType	Year		
Rice	Deeksy	ICT	D	KS4	Achieved	2014		
Rice	Becky	Spanish	C	KS4	Achieved	2014		
Rice	Becky	Maths	C	KS4	Achieved	2014		
Rice	Becky	English Language	B	KS4	Achieved	2014		
Rice	Becky	English Literature	B	KS4	Achieved	2014		
Rice	Becky	Food Technology	B	KS4	Achieved	2014		
Rice	Becky	English Language	A	KS2	Achieved	2014		
Rice	Becky	Maths	A	KS2	Achieved	2014		

Clearing the current selection

In order to clear the current data grid selection you can click the "Clear selection" button.

ClearSelection

Viewing Student's details

First of all you must select the “Student Details” button from the main menu form which is the same as the button opposite.



Filtering a subject/selecting a subject

In order to view a student's details you must select an attribute to filter student's by within the top left group box on the form (highlighted in green opposite). Once you have done this you must refine your filter option. Once you have done this you must click the search button, this will show all student's that match your filter in a data grid. An example can be seen in the picture opposite.

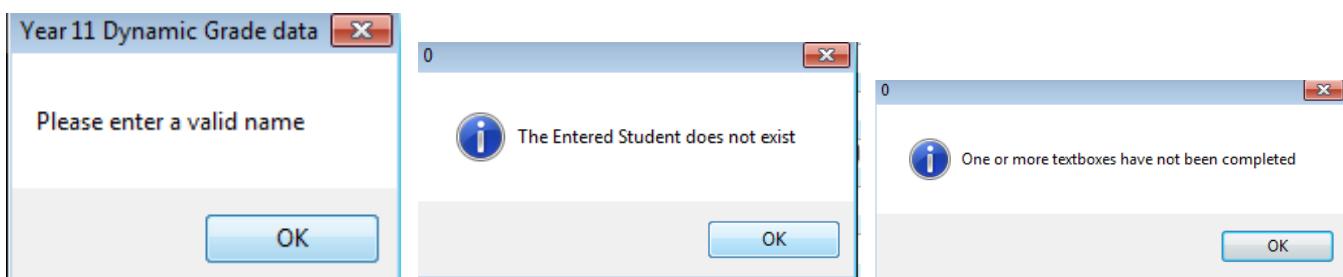
Firstname	Surname	Gender	FSM	SNS	Ethnicity	Attendance	DateOfBirth	Form	Year	Student Details
William	ZIMMERMANN	M	N	K	White - British	75.1	23/06/2000	11M	11	Edit Details
Will	WILLIAMSON	M	N	N	White - British	86.2	17/03/2000	11D	11	Edit Details
William	MILLWARD	M	Y	K	White and Black Caribbean	94.8	11/01/2000	11F	11	Edit Details

Editing a student's details

To edit a student's details you must click the “edit details” button which is on the same row as the student you want to edit the details of. You will then be presented with a separate form that will contain all of the details for the student you have selected. Once on this form you can edit any of the users' details using the entry fields and then clicking the “edit” button. Alternatively you can remove the student from the system by clicking the “delete” button.

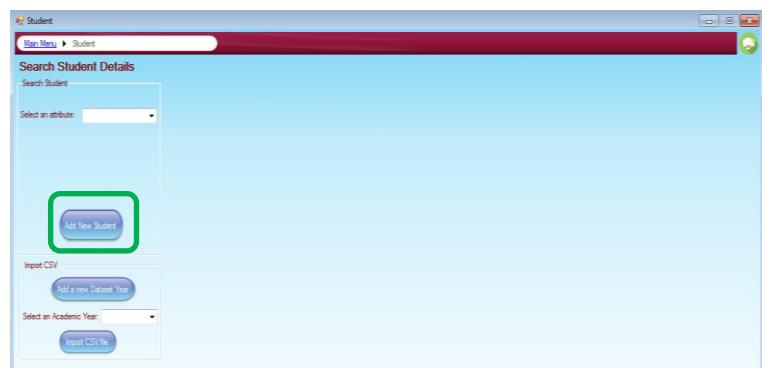
Examples of error messages

The following error messages are quite self-explanatory and you can ignore these messages by clicking the ok button and amending the correct input boxes.



Adding new students

In order to add an individual student to the system you must first be on the student details form. From here you need to click the “add new Student” button which looks like the one opposite, and is located within the group box on the left side of the form.



Adding a single student

Once you have clicked the “add new student” button a separate form will appear which looks like the one opposite. From here you need to complete all the entry fields on the form. Once you have completed all the entry fields with valid information you can insert the new record into the system by clicking the “add” button.

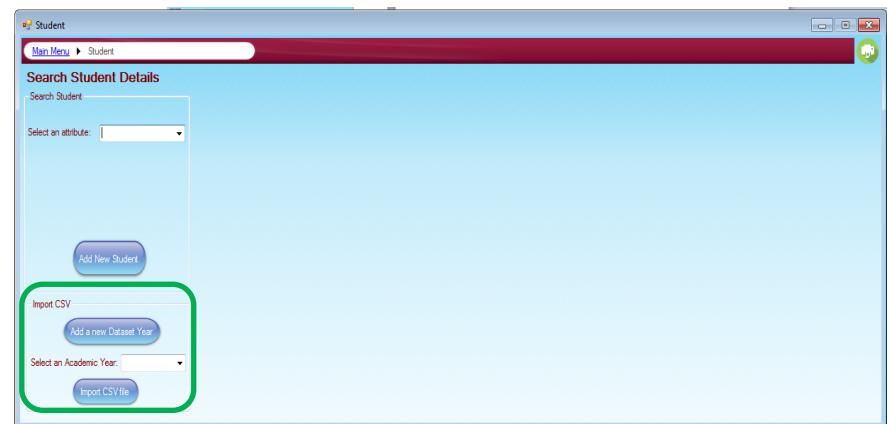
Adding multiple students

In order to add multiple students to the system in one instance you can import a report styled CSV. In order to do this you must first have a valid CSV file that contains student details. To do this you must use the report document that will be provided named “Import Student CSV”. This report will look like the example below.

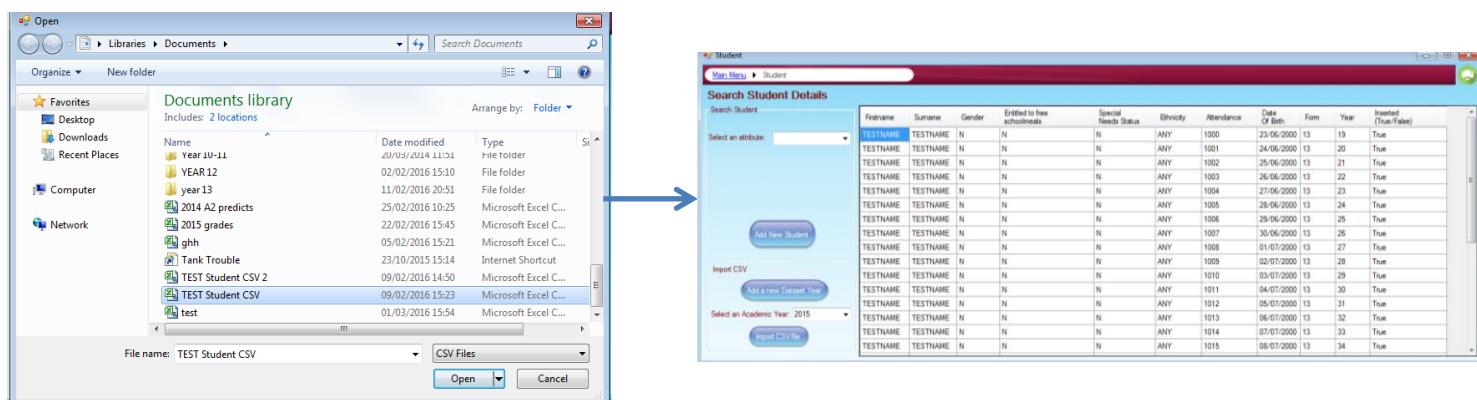
	A	B	C	D	E	F	G	H	I	J	K	L
1	Firstname	Surname	Gender	Entitled to free schoolmeals	Special Needs Status	Ethnicity	Attendance	Date Of Birth	Form	Year		
2												
3												
4												
5												
6												
7												
8												
9												
10												
11												
12												
13												
14												
15												
16												
17												
18												
19												
20												
21												
22												
23												
24												
25												
26												
27												
28												
29												
30												

Adding multiple students (continued)

Once you have completed a valid CSV report file you need to go to the student details form (instructions for this found above). Then you will need to select an academic year to assign the students to that you are importing to the system. You can do this by using the “select an academic year” combo box.

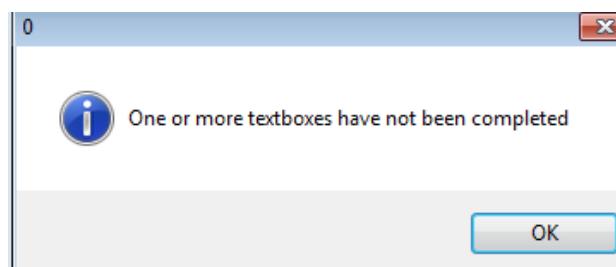


Once you have selected an academic year to import students to you need to click the “Import CSV file” button. This will open a file browser which will allow you to select your CSV report file. Once selected the records within the CSV file will be added to the system and confirmation of the records that you have imported will be visible in a data grid which will inform you which rows have or have not been inserted successfully. An example of this process can be seen below.



Examples of error messages (adding students)

The following error messages are quite self-explanatory and you can ignore these messages by clicking the ok button and amending the correct input boxes.



Viewing teachers within the system

First of all you must select the “Teacher Details” button from the main menu form which is the same as the button opposite.



Viewing Teacher's within the system

To search for teacher's that are a part of the current system, you need to select a filter attribute from the group box on the form named “search teacher”. Once you have selected an attribute to search by you can then filter by the attribute you have selected. To view the teachers that match your filter, you must click the “search” button. A data grid will then appear which contains all the teachers that match you search.

 A screenshot of a Windows-style application window titled "Teacher". The title bar includes "Main Menu" and "Teacher". The main area is titled "Search Teacher Details". It features a group box labeled "Search Teacher" with a dropdown menu set to "Attribute: Name". Below it is a text input field labeled "Name:" and a blue "Search" button. To the right is a data grid table with columns "Firstname", "Surname", and "Teacher Details". The table lists 20 teacher records, each with a "Edit Details" button. The "Search Teacher" group box is highlighted with a green oval.

Firstname	Surname	Teacher Details
Amelia	Lamberts	Edit Details
Amelia	Rutherford	Edit Details
Audrey	Carr	Edit Details
Andrew	Thomson	Edit Details
Anna	Miller	Edit Details
Audrey	Rutherford	Edit Details
Eva	Butler	Edit Details
Sella	Burgess	Edit Details
Benjamin	Knox	Edit Details
Boris	Wilson	Edit Details
Cameron	May	Edit Details
Cameron	Wallace	Edit Details
Carol	Fisher	Edit Details
Carolyn	Roberts	Edit Details
Charles	Carr	Edit Details
Christian	Lyman	Edit Details

Editing a teacher's details

In order to edit the details of a teacher within the system you first need to filter the teacher that you want to edit and then search for them. Then you need to click the “edit details” button which is on the same row as the teacher record that you want to edit. A separate form will appear which will allow you to edit the selected teacher's information. To confirm any editing changes you need to click the “edit” button. Alternatively you can delete the selected teacher from the system by clicking the “delete” button.

 A screenshot of a Windows-style application window titled "Add Teacher". The title bar includes "Main Menu" and "Teacher". Below it, a sub-header says "Edit Teacher Details". The form has two text input fields: "Firstname" containing "Amelia" and "Surname" containing "Lamberts". At the bottom are two blue buttons: "Edit" on the left and "Delete" on the right.

Adding a new teacher

To add anew teacher to the system you need to click the “add teacher” button which is on the “teacher details” form and within the group box named “search teacher”. Once you have clicked this button a separate form will appear which will allow you to add a new teacher. From here you can use the entry fields to add a new teacher. To confirm changes you need to click the “add” button.

 A screenshot of a Windows-style application window titled "Add Teacher". The title bar includes "Main Menu" and "Teacher". Below it, a sub-header says "Add Teacher Details". The form has two text input fields: "Firstname" and "Surname", both currently empty. At the bottom is a single blue "Add" button.

Student's classes

First of all you must select the “Students Classes” button from the main menu form which is the same as the button opposite.



Viewing a student's classes

First of all you must search for the student that you want to view the classes of. To do this you must use the “view a student’s classes” group box which is located in the left corner of the form. To search for a student you must select a filter attribute and then use some search criteria, once you have done this you need to click the “search” button within this group box. This will display a data grid that contains a list of all the student’s that match your search criteria. Next to select a student you need to click the “view classes” button which is on the same row as the student that you want to view the classes of.

ClassID	Fname	Surname	ClassGroup	Subject	Department	Year
91	James	Simon	Class Group 91	Business Studies	Technology	2014
94	James	Simon	Class Group 94	English Language	English	2014
99	James	Simon	Class Group 99	English Literature	English	2014
109	James	Simon	Class Group 109	History	Humanities	2014
112	James	Simon	Class Group 112	ICT	Technology	2014
120	James	Simon	Class Group 120	Maths	Maths	2014
126	James	Simon	Class Group 126	French	Languages	2014
135	James	Simon	Class Group 135	Computing	Technology	2014
141	James	Simon	Class Group 141	Chemistry	Science	2014
150	James	Simon	Class Group 150	Physics	Technology	2014

Adding a student to a class

First of all you need to search for the class that you want to add students to within the “add and remove students from classes” group box. To do this you need to select an attribute and then use search criteria to search for one or more classes. When you have done this click the “search” button within this group box. Once you have done this a data grid will appear and will contain a list of classes that match your search criteria. Then you need to click the “add student” button that is on the same row as the class you would like to add a student to. A separate form will now open which will allow you to do this. From here please refer to the “adding student’s to classes” section of the user manual.

ClassGroup	Subject	Department	Year	Fname	Surname	View Current Students	Add a Student to this Class
Class Group 107	Humanities	2014	Amelia	Lamberts		View Students	Add Student
Class Group 142	Chemistry	2014	Amelia	Lamberts		View Students	Add Student
Class Group 95	English Language	2014	Amelia	Rutherford		View Students	Add Student
Class Group 136	Computing	2014	Amelia	Rutherford		View Students	Add Student
Class Group 98	English Language	2014	Andrea	Car		View Students	Add Student
Class Group 101	English Literature	2014	Andrea	Car		View Students	Add Student
Class Group 103	Biology	2014	Andrea	Car		View Students	Add Student
Class Group 145	Chemistry	2014	Andrew	Thomson		View Students	Add Student
Class Group 155	Maths	2014	Anna	Miller		View Students	Add Student
Class Group 115	ICT	2014	Bella	Burgess		View Students	Add Student
Class Group 102	English Literature	2014	Benjamin	Knox		View Students	Add Student
Class Group 140	Biology	2014	Benjamin	Knox		View Students	Add Student
Class Group 123	Maths	2014	Boris	Wilson		View Students	Add Student
Class Group 119	Leisure and Tourism	2014	Cameron	May		View Students	Add Student
Class Group 108	Health And Social care	2014	Cameron	Wallace		View Students	Add Student
Class Group 122	Maths	2014	Carol	Fisher		View Students	Add Student
Class Group 105	Geography	2014	Ceryn	Roberts		View Students	Add Student

Viewing all students within a class

First of all you need to search for the class that you want to add students to within the “add and remove students from classes” group box. To do this you need to select an attribute and then use search criteria to search for one or more classes. When you have done this click the “search” button within this group box. Once you have done this a data grid will appear and will contain a list of classes that match your search criteria. Then you need to click the “view students” button that is on the same row as the class that you want to view a list of students of within that class. Once you have done this the data grid will refresh with a list of the students within your selected class.

The screenshot shows a Windows application window titled "Student Classes". At the top left is a dropdown menu labeled "Select an attribute: Name". Below it is a text input field "Name:" and a "Search" button. To the right is another dropdown menu labeled "Select an attribute: Class". Below it is a text input field "Class:" and a "Search" button. A green oval highlights the "Search" button under the "Class:" dropdown. To the right of these controls is a large data grid table with columns: Firstname, Surname, Form, ClassGroup, Subject, Department, Year, and "Remove this Student from this class". The table contains 15 rows of student data. The first few rows are: Abigail Sophie 12A Class Group 107 Geography Humanities 2014 Remove Student; Alicia Jesse 12E Class Group 107 Geography Humanities 2014 Remove Student; Amy Adam 12M Class Group 107 Geography Humanities 2014 Remove Student; Behan Locky 12M Class Group 107 Geography Humanities 2014 Remove Student; Caitlin Hannah 12J Class Group 107 Geography Humanities 2014 Remove Student; Charlotte Joe 12P Class Group 107 Geography Humanities 2014 Remove Student; Emily Joe 12A Class Group 107 Geography Humanities 2014 Remove Student; Emily Omari 12P Class Group 107 Geography Humanities 2014 Remove Student; Emma Benoit 12W Class Group 107 Geography Humanities 2014 Remove Student; Francesca Charlotte 12W Class Group 107 Geography Humanities 2014 Remove Student; Georgia Mabel 12N Class Group 107 Geography Humanities 2014 Remove Student; Hannah Tom 12B Class Group 107 Geography Humanities 2014 Remove Student; Holly Ben 12M Class Group 107 Geography Humanities 2014 Remove Student; Jessica Isobel 12G Class Group 107 Geography Humanities 2014 Remove Student; Katie Isabella 12A Class Group 107 Geography Humanities 2014 Remove Student; Kay Georgia 12G Class Group 107 Geography Humanities 2014 Remove Student.

Removing a student from a class

In order to remove a student from a class you first need to have selected a list of students for a class (look at the instructions above). Then you need to click the “remove student” button which is on the same row as the student you wish to remove from your selected class. A separate form will now appear which will allow you to do this. From here please refer to the “removing students from classes” section of the user manual.

The screenshot shows a dialog box titled "Remove Student From Class". At the top left is a breadcrumb navigation: "Main Menu > Student Classes > Remove Student". Below it is a section labeled "Current Selected Class:" with a table showing "ClassGroup: Class Group 107", "Subject: Geography", and "Year: 2014". Below this is a form with fields: "Student Firstname: Charlotte", "Student Surname: Joe", "Student Form: 12P", and a "Remove from Class" button.

Editing and adding classes

First of all you must select the “Teacher Classes” button from the main menu form which is the same as the button opposite.



Viewing Classes within the system

To view a selection of one or more classes you need to use the “search for a class” group box in the top left corner of the form. Once here you need to select an attribute which you will sue to filter your class selection. Next you need to provide valid search criteria for your filter option. Once this has been done you need to click the “search” button within this group box. A list of classes that match your search criteria will display in a data grid.

 A screenshot of a Windows application window titled "Teacher Classes". The window has a toolbar at the top with icons for back, forward, and search. Below the toolbar is a menu bar with "Main Menu" and "Teacher Classes". The main area contains a search panel on the left with a title "Search for a Class" and a dropdown menu set to "Teacher Name". It includes fields for "Name:" and a "Search" button. To the right of the search panel is a data grid table showing class details. The table has columns for Class Group, Subject, Department, Year, Firstname, Surname, and Class Details. Each row in the table represents a different class entry, such as "Class Group 107 Geography Humanities 2014 Amelia Lamberts Edit Class". At the bottom of the search panel is a "Create Class" button. The data grid shows multiple rows of similar data.

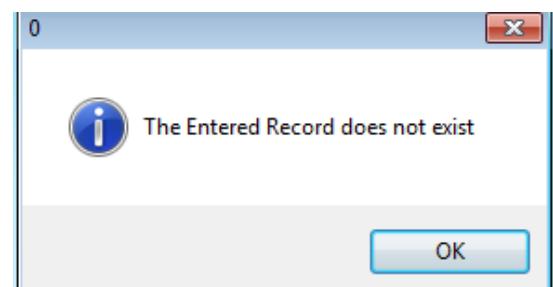
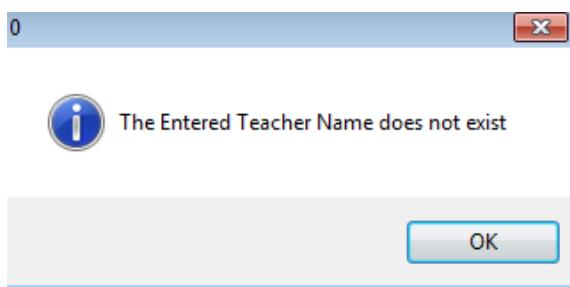
Editing a class within the system

First of all you must have a list of classes displayed in a data grid (follow the above instructions). Secondly you need to click the “edit class” button which is on the same row as the class that you want to edit. Once you have done this a separate form will display allowing you to edit your selected class. On this form you can use the text entry boxes to change the details of the class. To confirm any editing changes you need to click the “edit class” button. Alternatively to delete a class from the system click the “delete” button.

 A screenshot of a "Edit Classes" dialog box. At the top, it shows the navigation path: "Main Menu > Teacher Classes > Edit Classes". The main area is titled "Edit Classes". It contains five text input fields: "Teacher Firstname" (Andrea), "Teacher Surname" (Carr), "Class Group" (Class Group 139), "Subject" (Biology), and "Year" (2014). Below these fields are two buttons: "Edit Class" and "Delete".

Examples of error messages

The following error messages are quite self-explanatory and you can ignore these messages by clicking the ok button and amending the correct input boxes.



Adding a new class to the system

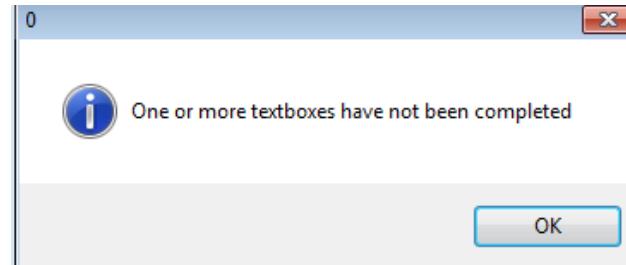
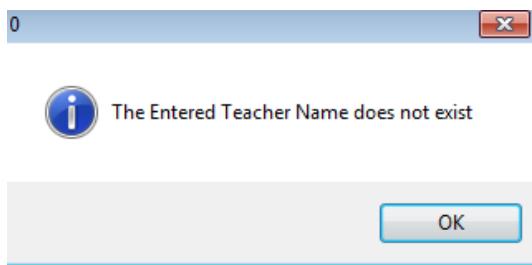
In order to add a new class to the system you must use the “create a new class” group box which is located in the bottom left corner of the “teacher classes” form. You must complete all the entry fields within this group box with valid data. Once you have done this, click the “create class” button to insert the new class record into the system.

The screenshot shows a Windows application window titled "Teacher Classes". At the top left is a "Main Menu" button and a "Teacher Classes" button. Below the menu is a search bar with dropdown menus for "Select an attribute" (set to "Teacher Name") and "Name". A "Search" button is next to the search bar. To the right of the search area is a table of existing class records. At the bottom left of the main area is a "Create a New Class" dialog box. This dialog has fields for "Teacher Fname", "Teacher Surname", "Class Group", "Subject", and "Year". A "Create Class" button is at the bottom right of the dialog. The entire "Create a New Class" dialog is highlighted with a green oval.

ClassGroup	Subject	Department	Year	Firstname	Surname	Class Details
Class Group 107	Geography	Humanities	2014	Amelia	Lamberts	Edit Class
Class Group 142	Chemistry	Science	2014	Amelia	Rutherford	Edit Class
Class Group 95	English Language	English	2014	Amelia	Rutherford	Edit Class
Class Group 136	Computing	Technology	2014	Amelia	Rutherford	Edit Class
Class Group 98	English Language	English	2014	Andrea	Carr	Edit Class
Class Group 101	English Literature	English	2014	Andrea	Carr	Edit Class
Class Group 139	Biology	Science	2014	Andrea	Carr	Edit Class
Class Group 145	Chemistry	Science	2014	Andrew	Thomson	Edit Class
Class Group 125	Maths	Maths	2014	Anna	Miller	Edit Class
Class Group 115	ICT	Technology	2014	Bella	Burgess	Edit Class
Class Group 102	English Literature	English	2014	Benjamin	Knox	Edit Class
Class Group 140	Biology	Science	2014	Benjamin	Knox	Edit Class
Class Group 123	Maths	Maths	2014	Boris	Wilson	Edit Class
Class Group 119	Leisure and Tourism	Humanities	2014	Cameron	May	Edit Class
Class Group 108	Health And Social care	Humanities	2014	Cameron	Wallace	Edit Class
Class Group 122	Maths	Maths	2014	Carol	Fisher	Edit Class
Class Group 105	Geography	Humanities	2014	Carolyn	Roberts	Edit Class

Examples of error messages

The following error messages are quite self-explanatory and you can ignore these messages by clicking the ok button and amending the correct input boxes.



Filtering Grades within the system

First of all you must select the “View Grades” button from the main menu form which is the same as the button opposite.



Filtering a selection of grades

In order to filter a selection of grades that are stored within the system you need to use the “view/edit a student’s grade” group box which is located in the top right corner of the form. From here you can use a multiple set of filters that act as cumulative filters to select a selection of grades. Once you are happy with your filter selections you need to click the “search” button. Once you have done this a list of grades that match your chosen search criteria should appear in the data grid.

Editing a grade

In order to edit a grade you need to click the “edit grade” button which is on the same row as the grade you want to edit. Once you do this a separate edit grade form will appear. From here please refer to the “editing grades” section of the user manual.

Firstname	Surname	Subject	Grade	Level	Grade Type	Year	Grade Details
James	Ma	Leisure and Tourism	F	KS4	Achieved	2014	Edit Grade
Richard	Madeleine	Maths	F	KS4	Achieved	2014	Edit Grade
Alexander	Christopher	Maths	F	KS4	Achieved	2014	Edit Grade
Hannah	Frank	Maths	F	KS4	Achieved	2014	Edit Grade
Ben	Maria	Maths	E	KS4	Achieved	2014	Edit Grade
Alice	Eve	Leisure and Tourism	E	KS4	Achieved	2014	Edit Grade
Oscar	Jasmine	ICT	E	KS4	Achieved	2014	Edit Grade
James	Ma	Maths	E	KS4	Achieved	2014	Edit Grade
Sean	Phys	Chemistry	E	KS4	Achieved	2014	Edit Grade
Alexander	Christopher	Chemistry	E	KS4	Achieved	2014	Edit Grade
Thomas	Owen	Physics	E	KS4	Achieved	2014	Edit Grade
Alexander	Christopher	Physics	E	KS4	Achieved	2014	Edit Grade
David	Thomas	Computing	E	KS4	Achieved	2014	Edit Grade
Thomas	Oliver	Geography	D	KS4	Achieved	2014	Edit Grade
Thomas	Brown	Geography	F	KS4	Achieved	2014	Edit Grade

Clearing the current selection

In order to clear the current filter selections you can click the “Clear selection” button.



View all Grades for a selected student

First of all you need to click the “Add a grade or view a student’s grade” button that is within the group box named “view/edit a student’s grade” on the view grades form. To get to this form follow the instructions previous.



Selecting a student

In order to view all the grades for a student you first must search for a student and then select that student. You can search for a student using the “select a student” combo box located in the top left corner of the form. From within here you need to select an attribute to filter your student selection by and then use valid search criteria. Finally you need to click the “search” button once you have done the above. A list of student’s which match your chosen search criteria will populate a data grid that will become visible. Next you need to click the “view grades” button on the same row as the student which you wish to view the grades for. An example of this process can be seen below.

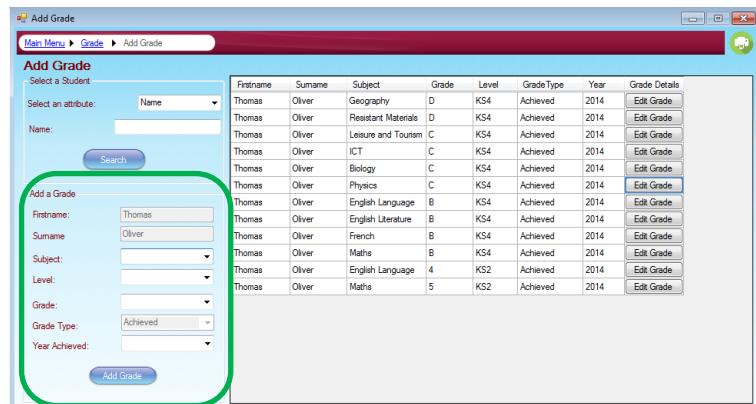
Firstname	Surname	Subject	Grade	Level	GradeType	Year	Grade Details
Thomas	Oliver	Geography	D	KS4	Achieved	2014	Edit Grade
Thomas	Oliver	Resistant Materials	D	KS4	Achieved	2014	Edit Grade
Thomas	Oliver	Leisure and Tourism	C	KS4	Achieved	2014	Edit Grade
Thomas	Oliver	ICT	C	KS4	Achieved	2014	Edit Grade
Thomas	Oliver	Biology	C	KS4	Achieved	2014	Edit Grade
Thomas	Oliver	Physics	C	KS4	Achieved	2014	Edit Grade
Thomas	Oliver	English Language	B	KS4	Achieved	2014	Edit Grade
Thomas	Oliver	English Literature	B	KS4	Achieved	2014	Edit Grade
Thomas	Oliver	French	B	KS4	Achieved	2014	Edit Grade
Thomas	Oliver	Maths	B	KS4	Achieved	2014	Edit Grade
Thomas	Oliver	English Language	4	KS2	Achieved	2014	Edit Grade
Thomas	Oliver	Maths	5	KS2	Achieved	2014	Edit Grade

Editing a grade

In order to edit a student’s grade you must first have selected a student you want to view all the grades for (instruction above). Next you need to click the “edit grade” button which is on the same row as the grade you wish to edit. From here please refer to the “editing grades” section of the user manual for more information.

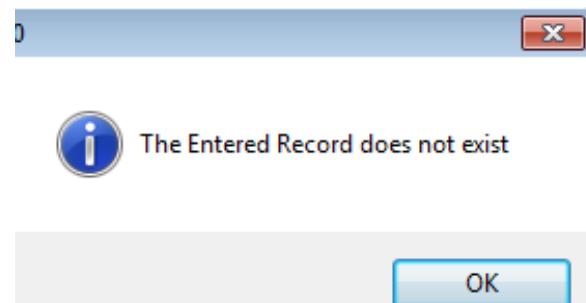
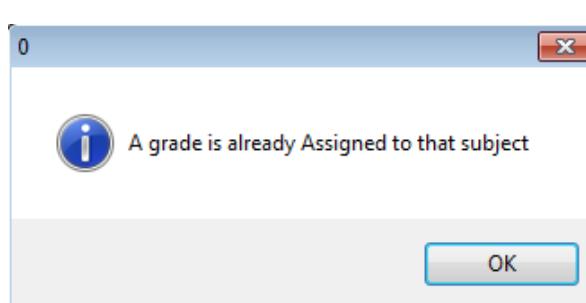
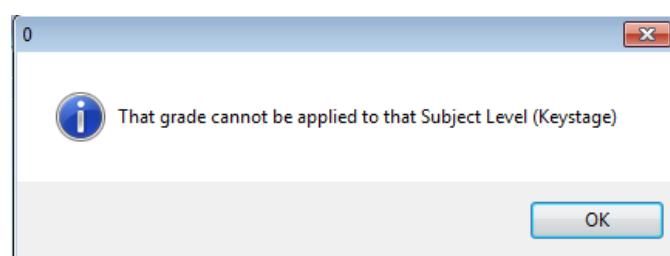
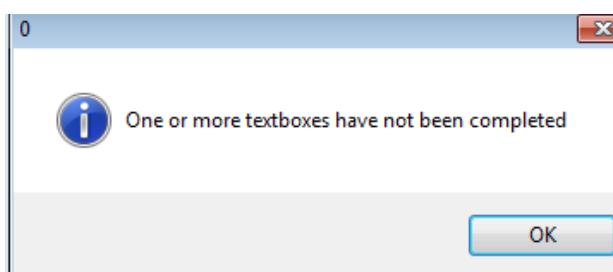
Adding a new grade for a selected student

In order to add a new grade to a selected student you need to use the “add a grade” group box which becomes visible after you have selected a student (see above for instructions). Then you must complete all the entry fields within this group box with valid data. Finally you need to click the “add grade” button.



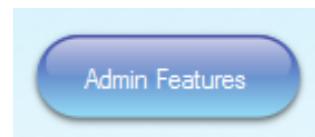
Examples of error messages

The following error messages are quite self-explanatory and you can ignore these messages by clicking the ok button and amending the correct input boxes.



Advanced admin features

First of all you must select the “Admin features” button from the main menu form which is the same as the button opposite.



Selecting an admin function

The admin functions that can be performed include:

- Add a new dataset year (academic year)
- Add a new Grade Type
- Add a new Subject
- Add a new level
- Add a new login
- Edit an existing Login

In order to select an admin function to perform you must use the “admin functions” group box which is located in the top left of the form. From here you need to select an admin action from the group box. Note that once you do this a data grid will become visible and will contain the records of all existing records that match your action along with a group box which will allow you to carry out your action. For example if you select the “add a new level” action, the data grid will contain a list of all the different level types the system stores.

LevelID	Level	Record Details
1	KS2	Delete Record
2	KS4	Delete Record

Deleting an admin record

If you wish to delete an admin record then you first of all need to select the appropriate admin action. For example if you wanted to delete a subject you would select the “add a new subject” action. Then from the data grid you would click the “delete record” which is on the same row as the subject you want to delete. Please note that deleting an admin record will most likely affect lots of records within the system.

Adding a new admin record

To add a new admin record then you first need to select the appropriate action (see above instructions). For example if I wanted to add the “KS5” level type to the system I would click the “add a new level” action. Then I would type the new level in the entry field of the group box and click the “add” button.

Frequently Asked Questions (FAQ)

Who should I contact if I have forgotten my login details?

If you have forgotten either of your username or password details you need to contact a system administrator, this could be either Mr Hewitt or Mr Nicoll. They will be able tell you your current login details or change your login details upon request.

I have accidentally deleted a record, is there any way to recover it?

If you have deleted a record and you remember the details of the record that you deleted you could try inserting the record back into the system using the instructions of this manual. Otherwise you should contact Mr Hardy or another ICT services technician who may be able to retrieve lost data from a backup copy of the system.

How do I uninstall the system?

If you wish to uninstall the system from your user account all you need to do is right click the executable file and use the file dialog to delete the program clicking the “delete” option.

Is there a way to reset a student's data?

This depends, if for example you need to change the academic year that a student is in then you can delete their student details from the system and then re-add them to the system but making sure to change the academic year. Otherwise, records that are attributed to a student such as grade records can only be deleted individually.

Can I use the program on a mobile or a tablet?

No, the system has been designed to work on either a laptop or desktop machine but you are free to experiment using it on a tablet device if you wish. For example the system will most definitely work on hybrid machines such as the Microsoft Surface that acts as a laptop and tablet.

Contact Details:

Contact:

Mr Hewitt or Mr Nicoll
Wirksworth Road,
Duffield,
Belper,
Derbyshire,
DE56 4GS,
01332 840645
RNicoll@ecclesbourne.derbyshire.sch.uk
DHewitt@ecclesbourne.derbyshire.sch.uk