

# BTEC DCT Database End User Project

Name – James Bunting

Candidate Number – 9428

Centre Number – 64395

Godalming College

## Contents

Project Analysis .....	4
Introduction .....	4
Current Issues .....	4
Research Methods .....	4
Interview .....	4
Questionnaire .....	6
Record Inspection .....	7
Observation.....	11
Summary .....	12
Variables List .....	13
Data Dictionary .....	14
Flow Charts .....	16
Use Case Diagram .....	22
Data Flow Diagram.....	23
Requirements List .....	24
Design.....	26
Overall System Design.....	27
Entity-Relationship Diagrams.....	28
User Interface .....	29
Data Structure.....	31
Storage .....	32
Processes.....	33
Validation .....	34
Testing Strategy .....	35
Code Dump .....	37
Home.vb.....	37
ViewStudInfofrm.vb.....	41
EditStudInfofrm.vb.....	48
ViewStudUnitsfrm.vb.....	52
CourseStudentsInfofrm.vb.....	55
ViewUnitsfrm.vb .....	60
ViewCourseInfofrm.vb.....	67
Technical Solution Details & Testing.....	73
Home.vb.....	74
Starting pop-up box .....	75

Home menu overview & database connection test .....	76
Navigation and Closing.....	78
ViewStudInfofrm.vb .....	80
Search by ID .....	80
Search by Unit .....	84
Home button.....	89
EditStudInfofrm.vb.....	90
Add/edit buttons.....	90
Like Pages.....	93
ViewStudUnitsfrm.vb .....	93
CourseStudentsInfofrm.vb.....	95
ViewUnitsfrm.vb .....	96
ViewcourseInfofrm.vb.....	98
Evaluation .....	99

## Project Analysis

### Introduction

The problem laid out to me by my end user, Caroline Miller – who is the head of department for BTEC DCT at Godalming College – is that their current database for BTEC DCT isn't satisfactory, and it can apparently be quite troublesome to use. What I have been tasked to do is to improve it based solely on comments from Caroline, an interview done with Caroline, notes on a discussion I had with BTEC students and my own programming skill.

BTEC DCT, more commonly known within the college as of now as IT, is a 100% coursework unit that can be taken as one of four awards; those being certificate (worth one AS-level), sub-diploma (worth one A-level), 90-credit (worth one-and-a-half A-levels), and diploma (worth two A-levels). There is also space to negotiate a full 180-credit extended diploma (worth three A-levels), but it is not open for students openly thus far. A database is an organised collection of data stored and accessed electronically from a computer system, and a database management system is software that interacts with end users, applications and the database itself to capture and analyse data inside the database. I am currently in my second year of a BTEC DCT diploma at Godalming College.

### Current Issues

Currently, the issues of tediousness are being resolved just by willpower. This apparently cannot be helped, and it can also lead to instances of misinformation. An example of this would be Caroline working through the database and inputting all the information, but accidentally putting the incorrect grade into one student's unit. If this were to happen, then not only would Caroline have to find the anomalous input and edit it herself, but she might also end up finding more anomalies – which could lead to a revision of the entire database. This misinformation could occur because another IT teacher may give incorrect information about a grade, or Caroline could misinterpret what was said.

Another issue with the current database is visibility of data for users other than DCT teachers.

### Research Methods

Interview – an interview with the 3<sup>rd</sup> party can be vital for obtaining information from a single user. If the user wishes to extend opinions, they are able to without constraint.

Observation – seeing how the current database is operated and what there is to be improved can prove useful indeed.

Questionnaire – a questionnaire can prove quite useful for gaining information from a group of people, about a specific topic.

Record Inspection – seeing what there currently is available to the 3<sup>rd</sup> party can definitely give me some ideas on what there is to improve upon according to the information I gather.

### Interview

I carried out an interview with the end user, Caroline Miller, so I could get an idea on what their current system does, what needs to be improved upon, and what needs to be added.

*James: "What would you say are the current problems you have with the database that you're using currently?"*

*Caroline: "Well, what comes to mind initially, is that there is no BTEC system for recording units taken*

*and grades achieved. The Self page used to at least show information about A-level courses, but never for a BTEC”*

*James: “Alright. How do you feel this has affected you for logging and retrieving information?”*

*Caroline: “Oh, it’s not so bad for myself, as I’m more than used to what I’m using. However, for everyone else involved – such as other departments – this can be a real challenge.”*

*James: “How would it be a challenge?”*

*Caroline: “Well, over all the departments, there’s a real amount of inconsistency. Every department seems to have different methods or systems to record everything, and some departments refuse to enter work into multiple areas. Entering information into Godalming Online directly is one thing, but then also having to enter all that data into a spreadsheet is something most departments might not want to do.”*

*James: “Okay. How do you currently deal with sorting information?”*

*Caroline: “Right now, everything I record is put directly onto Godalming Online. The grades for controlled assessments are recorded as pass, merits, distinctions and fails, and each unit is delved into individually too. The times of the controlled assessments are recorded too, and everything stored on Godalming Online is visible to anyone – students and teachers. This is not only so I can see where a student would need to improve and be able to point it out for them, but also so that student can see for themselves – which comes in handy when resubmission time rolls around. It would also be handy, provided other departments actually recorded grades like this.”*

*James: “Mhm. So, are there any crucial areas for improvement?”*

*Caroline: “Oh, of course. Gathering the information and individually inputting it into Godalming Online one item at a time can be very tedious and time-consuming. This process should most definitely be streamlined. Also, making sure data is easy to access is imperative, and copying it should be simple as well. Re-writing entire spreadsheets worth of data takes up a lot of time, so being able to clearly see information without having to do so much, and then being able to copy it should be a priority.”*

## Questionnaire

I wrote out a questionnaire for DCT students on how they found the current methods of seeing their own data. Below are responses from two students, **Niall Forth-Gibbons (upper sixth DCT diploma student)** and **Malin Cory (upper sixth DCT diploma student)**. Note that the colour key relates to the two students who answered the questions.

Question 1: Currently, how do you access information about yourself, such as grades?

**Godalming Online.**

**The Self page, and Godalming Online.**

Question 2: How helpful do you find this information?

**For working through feedback, it's okay.**

**It can be helpful, but it's still not great considering everything's in different places.**

Question 3: Would you say it could be easier to view?

**Yes.**

**Yes, the information should be all in one place.**

Question 4: What would be the most important change for you, provided you want a change?

**Being able to easily figure out my predicted grade instead of going to an external site.**

**Having all the information in one easy to reach place would be nice. Having to go to separate areas for different pieces of info isn't good.**

Question 5: Is there anything you *wouldn't* want if a new info viewing system were implemented?

**No.**

**Having too much information would be bad.**

## Record Inspection

The following blocks of information under this heading have within them screenshots from services and systems used to record and store data about students and their grades.

Module	Grade	Credit
Module 1:	D	10
Module 2:	D	10
Module 3:	D	10
Module 4:	D	10
Module 5:	D	10
Module 6:	D	10
Module 7:	D	10
Module 8:	D	10
Module 9:	D	10
Module 10:	D	10
Module 11:	M	10
Module 12:	M	10

The two above images are screenshots from the webpage used for working out current grades, and predicted grades. This part of the webpage is a couple of scrolls away, and the link is very obscure – I had to ask Caroline to give me the link to the site because it isn't always available on Godalming Online. Anyway, I input my current award and how many modules I was taking overall into the areas seen in the left image, and then the menu seen in the right image appeared. Each unit is worth 10 credit, so that was simply left as is. As for grades, I had to type each one individually which took a fair bit of time. Once that was done, I filled out the remaining modules that I hadn't yet done to find a minimum grade for scoring the highest possible grade...

**Your overall grade is**

**D\*D\***

**this translates to 1060 QCF points**

**This takes you to a total of 280 UCAS points**

...And this is what appeared to the right of the menu. It told me how many points the grade was worth as well as what the grade is. So my predicted grade area should show information like this.

Summary
U6 Choices Exam Results Markbook

Attendance: 94.16 %  
Punctuality: 100.00 %  
GCSE average: 5.33

Status: **Enrolled**

Student No.: <18XXXX>  
Forename: <Student>  
Pref. name: <Stud>  
Middle Name(s): <Ent>  
Surname: <Dimitris>  
DOB: <DD/MM/YYYY>  
Gender: <Gender>

Address: <Address 1>  
<Address 2, Town>  
<County, Postcode>

Home No: <01483 XXXXX>  
Mobile No: <07500 XXXXX>  
Email: studentent@gmail.com

Tutor year: <19/19 (L) 19/20 (L)>  
Tutor Group: 8H  
Tutor: <Tutor Minimus>  
Senior Tutor: <Tutor Maximus>

Medical: No  
Allergies: No

School: <Secondary School>  
Interview: <DD/MM/YYYY>  
Interviewer: <Tom Interv>  
Print balance: £9.78  
College Bus: No  
Gym induction: No

Status on 06/11/2019 ▼ : N/A D-Level: N/A Cause For Concern: N/A

Attendance summary for week: **Mon 04/11/19 to Fri 08/11/19** [View absences/exemptions](#) [Historical attendance](#)

Start	Monday	Tuesday	Wednesday	Thursday	Friday	End
14/10/19	P P P P P	N N R R R	P P P	P P P	P P P P P	18/10/19
21/10/19	P P P P P	P P P P P	P P P	P P P	P P P P P	25/10/19
28/10/19						01/11/19
04/11/19	P P P P P	P P P P P	N N N	N N N		08/11/19

**Key:** P present L late (<5 mins) R late (5 mins+) X not needed T trip/exam R reported authorised absence  
A unreported unauthorised absence U reported unauthorised absence H holiday Q student absence queried N register not yet taken

**Parents/Guardians Living With Student**

Custody: No  
Relation: Mother  
Name: Mrs Jules Bunting  
Mobile: 07775 537058  
Work No:  
Email: julesbunting@btinternet.com

18/19 (L) 19/20 (L)

	08:45 - 09:30	09:30 - 10:15	10:15 - 10:45	10:45 - 11:30	11:30 - 12:15	12:15 - 13:00	13:00 - 13:15	13:15 - 14:00	14:00 - 14:45	14:45 - 15:30	15:30 - 16:15	16:15 - 17:00
Mon		PTC:8H AYJ 406 100.00 %	Break	INH:M1 CXM 136 100.00 %	INH:M1 CXM 136 100.00 %	SSN:D3 PZJ 235 100.00 %	Lunch	Lunch		INH:M1 CXM 136 100.00 %	INH:M1 CXM 136 100.00 %	
Tue	CPB:E1 JMH 135 75.00 %	CPB:E1 JMH 135 75.00 %	Break	INH:M1 CXM 136 71.43 %	INH:M1 CXM 136 71.43 %		Lunch	Lunch		LP2:G1 NONE ---		
Wed			Break	CPB:E1 JMH 135 100.00 %	CPB:E1 JMH 135 100.00 %	LP3:G3 NONE ---	Lunch					
Thu	INH:M1 EJE 136 100.00 %	INH:M1 EJE 136 100.00 %	Break		LP1:A6 NONE ---	INH:M1 EJE 136 100.00 %	Lunch	Lunch		INH:M1 EJE 136 100.00 %		
Fri			Break			CPB:E1 PJM 135 100.00 %	Lunch		CPB:E1 PJM 135 100.00 %	INH:M1 CXM 136 100.00 %	INH:M1 CXM 136 100.00 %	

Reviews: [Show review](#)

University choices: [Show form](#)

Filter [Clear all](#)

Filter by type... Filter by level... Filter by status... Filter by unread... Filter by course...

Created	Updated	Type	Level	Updated by
25/03/19 16:20	28/06/19 07:55	121 Information Technology	Head of Department	CXM/CXM
31/10/18 11:33	12/06/19 11:43	121 Computer Science	Subject Tutor	PJM/PJM
07/02/19 11:42	07/02/19 11:42	Personal Tutor 121	Personal Tutor	EJE/EJE

Course	Class	Subject	Course start/end	Teachers	Attendance	Latest Target	B3	ARG	Pred grade	Grade
1;SSN	D3	Study Support	14/10/19 - present	PZJ	100.00 %					
2;LP1	A6	Library Period	06/09/19 - present		N/A					
2;LP2	G1	Library Period	06/09/19 - present		N/A					
2;LP3	G3	Library Period	06/09/19 - present		N/A					
4;CPB	E1	Computer Science - A level	06/09/19 - present	JMH PJM	90.48 %				C	
4;INH	M1	Information Technology - (IT) BTEC Diploma (2 A-level equivalent)	06/09/19 - present	EJE CXM	95.24 %				DD*	
4;PTC	8H	Personal Tutor	06/09/19 - present	AYJ	100.00 %					

**GCSE results** Average score: 5.33

Date	Qualification	Grade
Jun 2018	AQA - GCSE English Language	5
Jun 2018	AQA - GCSE English Literature	5
Jun 2018	AQA - GCSE in Computer Science	5
Jun 2018	AQA - GCSE in Drama	3
Jun 2018	AQA - GCSE Mathematics (H)	5
Jun 2018	AQA - GCSE Science: Biology Tier H	4
Jun 2018	AQA - GCSE Science: Chemistry Tier H	5
Jun 2018	AQA - GCSE Science: Physics Tier H	6
Jun 2018	AQA - Short-course GCSE Religious Studies	3
Jun 2018	OCR - GCSE in Geography B	4

The above two images are taken from the Self page (some items have been artistically recreated to avoid leaking private information). Firstly, the top image shows all the information about a student,



such as their name (and preferred name), attendance, address, etc. Of course, some of this information isn't completely necessary for the DCT-specific database, but some of it could still remain available so links to a student can easily be made. Next up, viewing one's courses and grades on the Self page is quite difficult if you're in a rush. One must scroll down a bit to reach the course table, then focus their eyes onto the very right of the table to see one grade. That's it. Just one grade for all of the course. No break-down at all. The Self page does have good information on a student, but this being the place students will go to see information most frequently, they may not know their grades in certain units, and have no way to find out because of how the Self page is.

Grade item	Calculated weight	Grade	Percentage	Feedback	Contribution to course total
<b>Summary</b>					
Unit 1 grade	-	DIST	100 %	Resubmission Assessment - 18/6/19 Unit 1: DIST Caroline Controlled Assessment - 4/6/19 Unit 1: PASS Caroline	-
Unit 2 grade	-	DIST	100 %	Controlled Assessment - 1/3/18 Unit 2: DIST Caroline	-
Unit 17 grade	-	DIST	100 %	Resubmission Assessment - 2/9/19 Unit 17: DISTINCTION Caroline Controlled Assessment - 7/7/19 Unit 17: FAIL Caroline	-
Unit 20 grade	-	DIST	100 %	Resubmission Assessment - 18/6/19 Unit 20: DISTINCTION Caroline Controlled Assessment - 4/6/19 Unit 20: FAIL Caroline	-
Unit 22 grade	-	DIST	100 %	Controlled Assessment - 1/3/18 Unit 22: DIST Caroline	-
Unit 28 grade	-	DIST	100 %	Resubmission Assessment - 18/6/19 Unit 28: DISTINCTION Caroline Controlled Assessment - 4/6/19 Unit 28: FAIL Caroline	-
Unit 30 grade	-	-	-		-
Unit 31 grade	-	-	-		-
Unit 36 Grade	-	DIST	100 %	Controlled Assessment - 1/3/18 Unit 36: DIST Caroline	-
Unit 37 grade	-	-	-		-
Unit 40 grade	-	DIST	100 %	Controlled Assessment - 7/7/19 Unit 40: DISTINCTION Caroline	-

Fortunately, there is at least *something* for students to view their grades. The above image shows each unit the student is taking, along with the grade they got, when they got it, and under what circumstance – i.e. controlled assessment, resubmission assessment, or reserve assessment. Not only that, but if the student scrolls down enough, they will see...

<b>Unit 1</b>					
U1-P1	0.00 %	Achieved	100 %		0 %
U1-P1	0.00 %	Achieved	100 %		0 %
U1-P2	0.00 %	Achieved	100 %		0 %
U1-P3	0.00 %	Achieved	100 %		0 %
U1-P4	0.00 %	Achieved	100 %		0 %
U1-P5	0.00 %	Achieved	100 %		0 %
U1-P6	0.00 %	Achieved	100 %		0 %
U1-P7	0.00 %	Achieved	100 %		0 %
U1-P8	0.00 %	Achieved	100 %		0 %
U1-M1	0.00 %	Achieved	100 %		0 %
U1-M2	0.00 %	Achieved	100 %		0 %
U1-M3	0.00 %	Achieved	100 %		0 %
U1-D1	0.00 %	Achieved	100 %		0 %
U1-D2	0.00 %	Achieved	100 %		0 %

...the information about all the criteria within a unit. This way, a student can see the details on any given unit that they're taking. They can see whether they achieved the criteria or not within a unit, and this can help them if they missed out on a high grade in a unit and need to resubmit. This all tells

me that my database would need to have a good amount of information about any student's particular grades on a unit.

## Observation

From observing the current method of how data is noted, stored and relayed to other personnel, I can give a run-down of how it works.

- To begin, the head of department Caroline writes down the grades into Godalming Online (once it's marked of course, that's its own problem). All the different fields are individually filled in by Caroline by hand, which takes a lot of time.
- Once this is done, however, the unit grades are available for viewing by the student they're related to. For instance, I can see my feedback, but I can't see someone else's.
- Navigating to this area is quite troublesome, though, since it's in an obscure place.
- As for the predicted grade calculator, the way that works is the grades are input into an external website not run by the college. The website takes the grades input to work out the current grade, and simply filling the rest with distinctions can give the highest possible predicted grade.
- For inputting information into the Self page, Caroline heads to each individual person and writes in the predicted grade.
- For exporting the data, there is no easy or simple way. It all has to be put into a separate spreadsheet individually, by hand.

## Summary

To summarise, the current database is operational, but still could do with refinement. Technically, it's not even a database system, but rather a collection of various areas for storing data. What would be good is to bring the different key areas of these data stores into one area for convenient viewing and editing. On top of that, the current system would need a way for students and teachers alike to predict grades easily, rather than using a silly external site. Finally, the most imperative part would be to implement an easy way to export data, such as sending it to a file to be copied, or having it copied to the clipboard from the page.

## Variables List

- User – Head of Department
  - The head of department should and will have full access to the database, and (optionally) be able to edit all of the database outside of the display, such as making additions and removing information.
- User – Student
  - The student should only be able to view the database, without being able to make any edits to it.
  - They also do not need to see anything about other people, just themselves.
- User – Teacher
  - Teachers have similar access permissions to students, only being able to view information.
  - However, teachers would want access to every student's information, instead of just a singular student.
- User – Database Manager
  - The database manager, like the head of department, should have full access to the database and be able to make additions to the database.
- Item – the Database
  - The database would have information about students, the units, the students' grades on the courses, and have links to everything within these.

## Data Dictionary

Student table:

Field Name	Field Purpose	Field Type	Field Size	Example Data	Validation
Forename	Store the student's forename	String	2-15	"James"	Text
Surname	Store the student's surname	String	2-15	"Bunting"	Text
Student ID <b>(Primary Key)</b>	Store the student's unique ID	Integer	6	189428	Numbers
Tutor Year	Stores the student's tutor year	String	2	"U6"	"L6" or "U6"
Tutor Group	Stores the student's tutor group code	String	2	"8H"	Number, followed by character
IT Course Code	Stores the student's course code	String	5	4;INH	Text
Predicted Grade	Stores the student's overall predicted grade	String	1-6	D*D*	Text

Unit table:

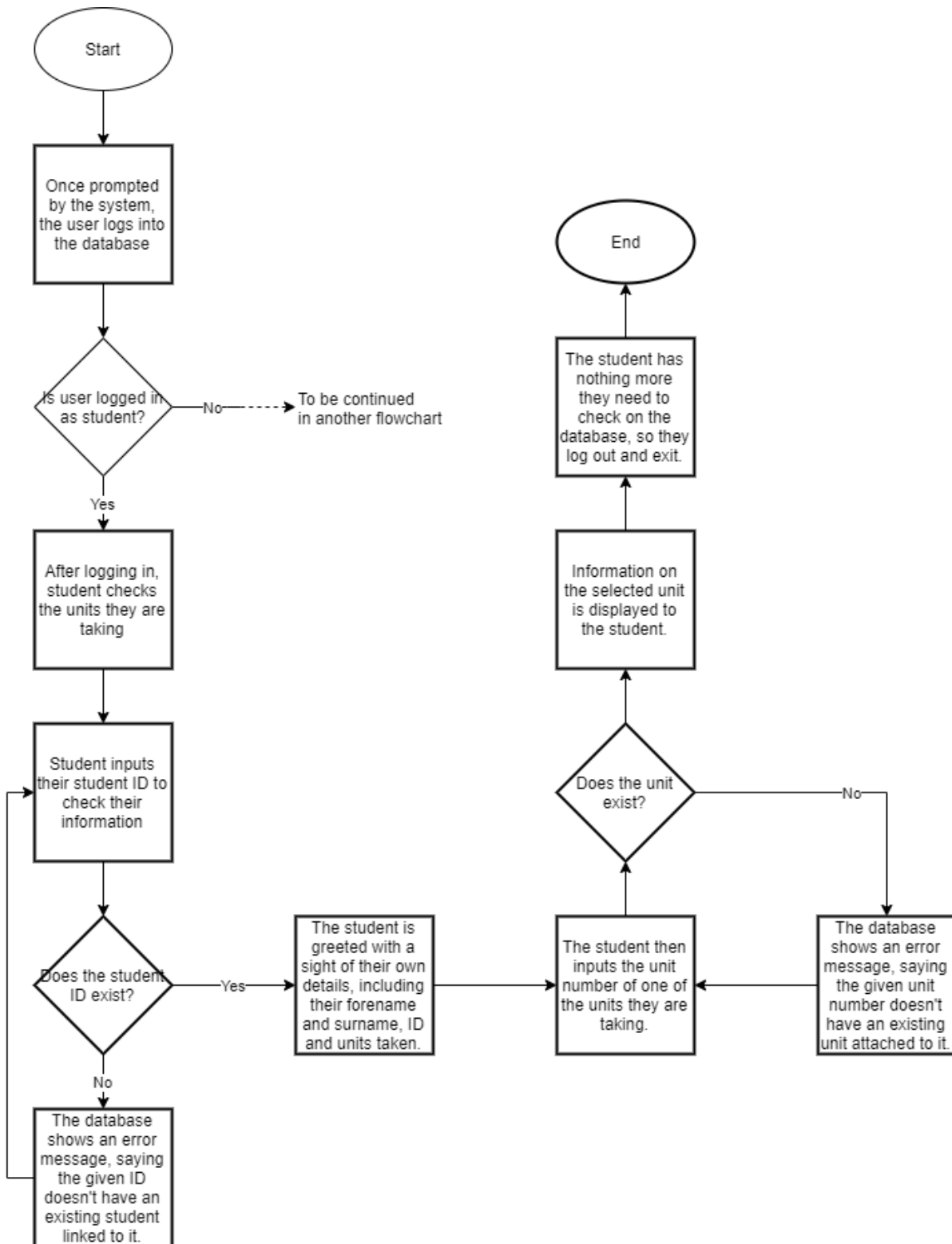
Field Name	Field Purpose	Field Type	Field Size	Example Data	Validation
Unit Number <b>(Primary Key)</b>	Stores the unit's unique number	Integer	1-2	37	Number between 1 and 43
Unit Name	Stores the unit's name	String	1-99	"Installing & Upgrading Software"	Text
Unit Type	Lists whether a unit is mandatory, optional, or specialist optional	String	3	"CMP"	"CMP", "OPT", or "SPE"

Grades (student-unit link) table:

Field Name	Field Purpose	Field Type	Field Size	Example Data	Validation
------------	---------------	------------	------------	--------------	------------

Student ID <b>(Foreign Key)</b>	Stores the student's unique ID	Integer	6	189428	Numbers
Unit Number <b>(Foreign Key)</b>	Stores the unit's unique number	Integer	1-2	37	Numbers between 1 and 43
Unit Grade	Stores the grade the student achieved for the unit	String	1-2	D*	"P", "M", "D", "D*", or "F"

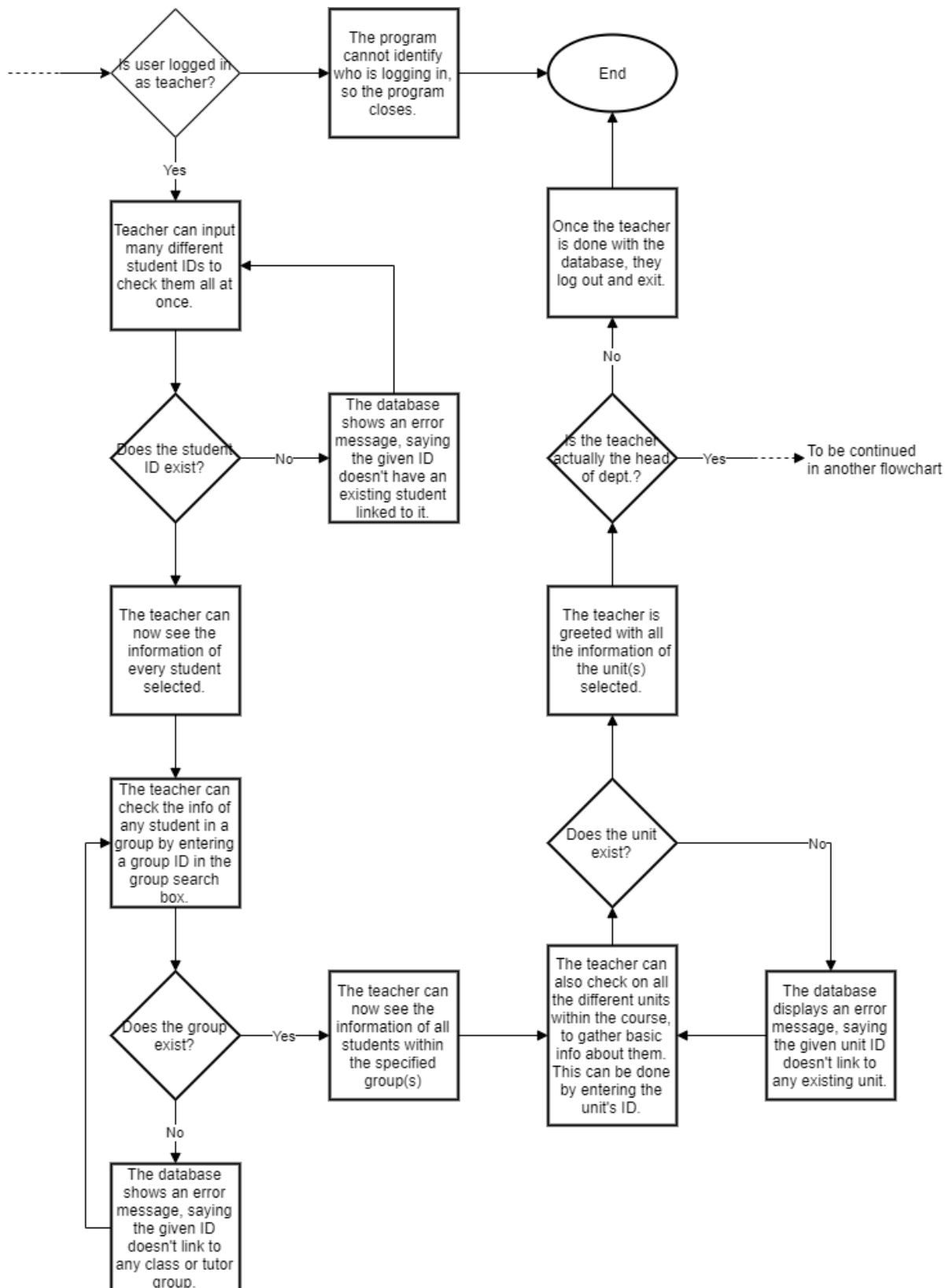
## Flow Charts



This first flowchart goes through the rhythms of a student logging into the database and finding out information about themselves. It highlights parts of the processes, such as logging in. To go into further detail, the student would be taken to a user interface for logging in once the system is booted up. After validating that the student is in fact a student, the student goes through the system's buttons to end up at the database's query page. Here, if the student inputs their ID, they can check their own information and – if they asked to see it – they can also see their grades on the

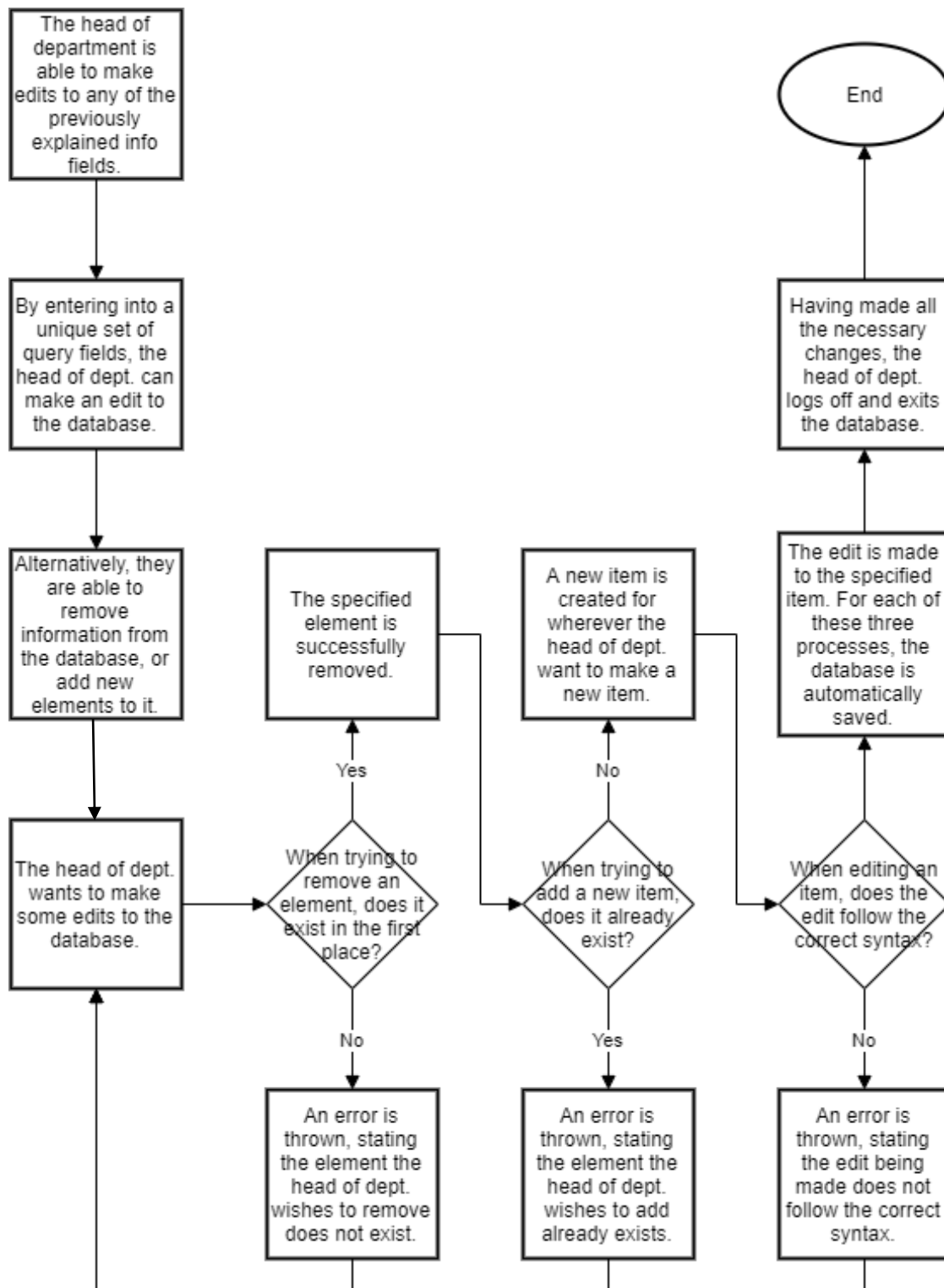


units. Furthermore, the student information viewing is validated by comparing their ID used for logging in with the ID input into the system. If the two don't match, the student won't see anything. If they put their own ID in though, they will be able to see their information. Next up, the student wants to check up more information on a specific unit. They simply back out to the home page, before clicking through the labelled buttons to reach the unit-specific query page. Here, they input the unit number and are greeted with information about that unit. On top of that, if they are taking that unit, they can see their current grade as well – they'd have to input their student ID as well to see it. Once they are done using the database, they simply log out of it via a button on the home page. Obviously, students are completely unable to make any edits to the database, and this is validated by the initial login.



This second flowchart (or rather, the second part of the flowchart) goes through the processes of a teacher using the database. When they log in, they can see a similar user interface to that of the student, but they instead can see information on the full tables *and* individual students if they want. This teacher wants to view information on a class or tutor group, so they go through the buttons to view the query page. From here, they can input the tutor group ID in the tutor group field, *or* the

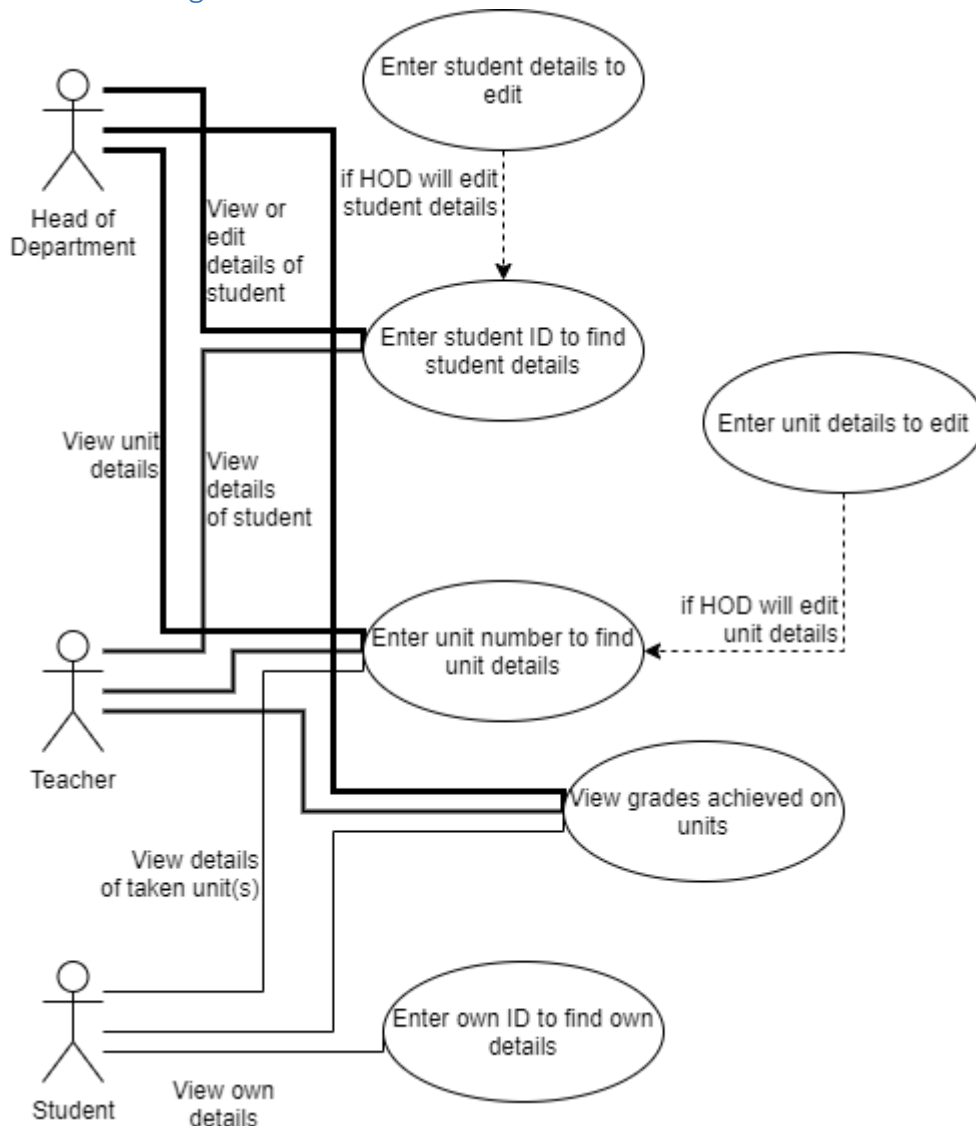
class ID into the class field. Once done, every student in that tutor group or class will have their information visible to that teacher. They can check what units they are doing, what award they are taking, and (optionally) their grades too. If a teacher wants to know more about a unit, they can input the unit ID much like how the student did, and the information on the unit will become available to them. Once they are done using the database, they simply log out of it via a button on the home page. Though teachers can see all the information, they cannot make edits to any of it. That privilege is reserved for the head of department.



This here third flowchart reflects some of what the head of department could do with the database. Once logging in, their unique login details would tell the database that the head of department has logged in, and they will be greeted with a rather different user interface. On top of the usual “view info on student(s)” or “view info on unit(s)” buttons, the head of department also has access to a few buttons used for adding elements, making edits to elements, and also archiving pieces of data. First off, the head of department wants to make an edit to the database – say for instance, a student’s name was typed incorrectly and needs changing. The head of department would first input

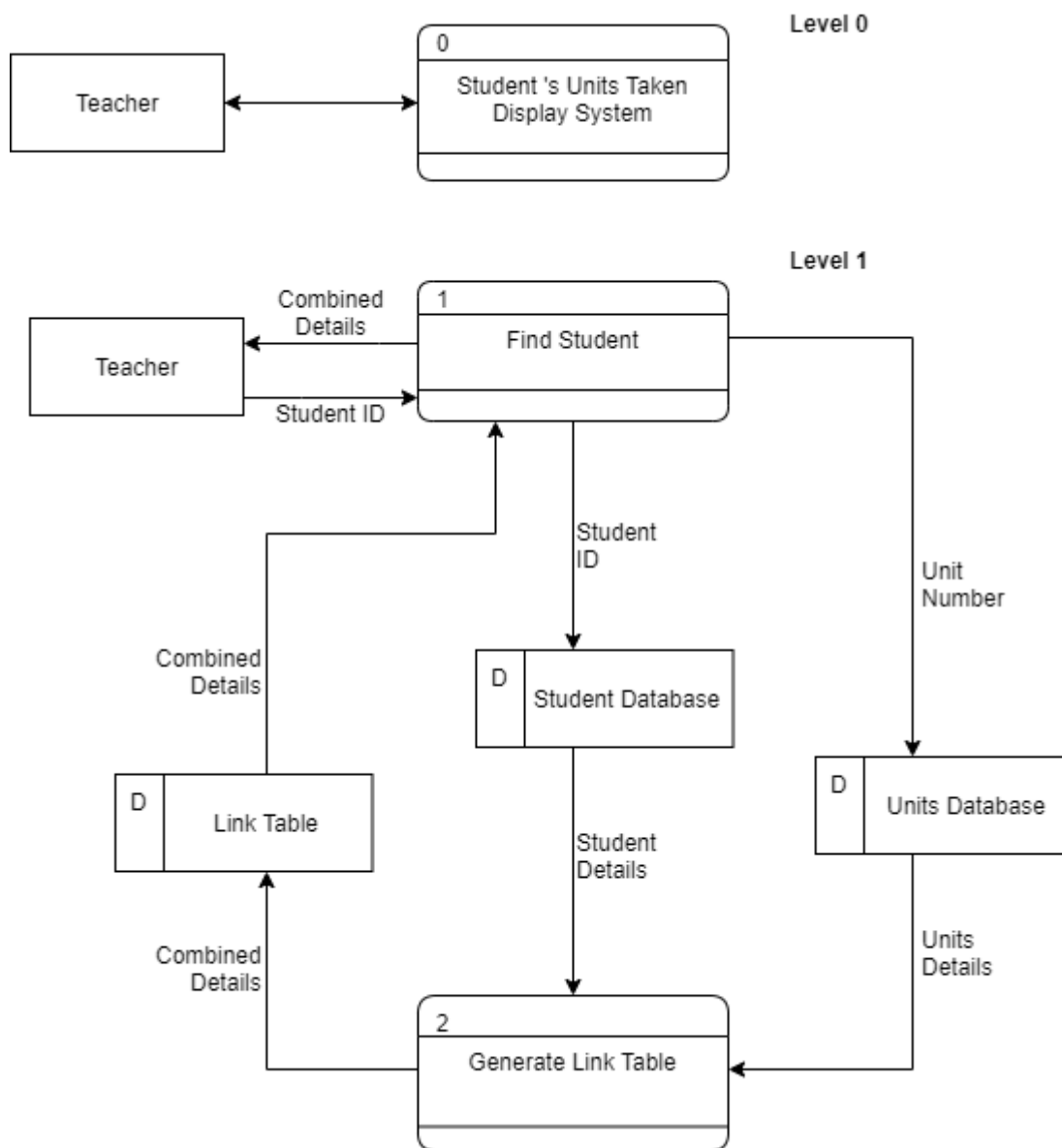
the ID of the student being edited, before writing into the edit field what column this falls under – the head of department would be informed of all the table names, and table column names as a small part of training. After declaring what field is to be changed (in this case, the Forename), the new name is input in the following input box and the changes are made once the “Make Edit” button is pressed. Alternatively, the head of department may want to add a new student to the database. This is done from an entirely different page in the edit section. From the edit section, the head of department goes to the “Add Item” button and then presses the button with the right table name (Student, Unit, or Grade). The head of department then fills out all the (known) information into the table – any blank information is set a placeholder value as to not crash the database. As for archiving items within the table, the head of department simply inputs the ID of what they are removing. Say, for instance, a student is no longer taking the course. The head of department inputs their student ID, and the items linked to that ID are all archived – this includes their information on the Student table, and the items in the Grade table. All of these processes are validates as to avoid crashes.

## Use Case Diagram



The above case diagram portrays the various interactions between a student; or a teacher; or the head of the department, and the database. In the case of the student, it is shown that they cannot access information on other students, only their own information can be checked. They can, however, check information on any unit. On top of this, the student is also able to check their own grades on units they are taking (again, only their own info, not other students' info). In the case of a teacher, they are able to view information the student can see, but they can see information on any student. They are able to view specific groups (classes, tutor groups), or view every student's information. As with the student, they can also see unit details and grades students have achieved. In the case of the head of department, they can view anything the teacher can view, and also make edits to the information. On top of this, they are also able to archive data that isn't necessary (though the use case diagram does not mention this, it is eluded to in the above diagrams).

## Data Flow Diagram



This above data flow diagram shows the process of a teacher accessing the database to see what units the student is taking. First, the teacher enters the student ID. Afterwards, the student database is accessed and, via the student ID, the database is accessed and the student's information is accessed. Amongst this information is the units they're taking. The unit numbers are taken and, from the database of units, the details of the units are accessed. Following this, the units' information and the student's information are combined to display all the necessary information in a link table, and this information is displayed to the teacher. Though the table diagram doesn't say, the head of department or the database manager could use this information obtained and make any additions to the details of the student(s) or the unit(s).

## Requirements List

To design a solution, it is essential to know the end goal that is trying to be achieved. Here, I have broken down my goal into a selection of requirements. Put simply, if I meet these requirements, I will have effectively made a solution to the problem.

1. To be easy to use.
  - 1.1. The end user must be able to navigate to any table of information or any function in the database application, within five mouse clicks.
  - 1.2. Inputs must absolutely be validated to avoid crashes, and the end user must be informed of an invalid input when one is detected.
  - 1.3. The user interface must be easily understood by anyone – not just someone familiar with IT.
2. To be able to store student data.
  - 2.1. The end user will have at their disposal a database that includes a table for student data
  - 2.2. The end user should be able to enter a student's details as a table item through a user interface, adding it to the table.
  - 2.3. The interface must be validated to ensure all of the data is correctly typed, and that no fields are missed.
3. To be able to store information about units.
  - 3.1. The end user will also have within this database a table for information on units.
  - 3.2. The end user should be able to enter information about new units that have been added through a user interface.
  - 3.3. The user interface for this table will be separate from the user interface for adding data to the student table.
  - 3.4. Much like adding data to the student table, adding data to the unit table must be validated.
4. To be able to store students' grades.
  - 4.1. The end user will have available to them a table of students connected to units they're taking, and what grades they've achieved.
  - 4.2. These unit grades would be easily updated, added, deleted, etc. if need be.
  - 4.3. These unit grades must have validation to ensure only grades can be typed into them.
5. To be able to view any of the above information.
  - 5.1. With a few simple clicks onto the user interface, the end user should be able to reach a query page to view information about a student (and their grades), or information on a unit.
  - 5.2. The queries would show information on the student table row just by inputting the student ID number.
  - 5.3. The queries would show information on the whole unit (including a description) just by inputting the unit number.
6. To be able to edit any of the above information.
  - 6.1. A student might change their name, tutor group, award etc., and this will need to be updated on the database.
  - 6.2. A unit might become available, or be removed from the course, so this would need to be updated on the database.
7. Any information must be achievable.
  - 7.1. To save space in the database, information on students that have already left should be able to be archived.
  - 7.2. The data should **not** be deleted as opposed to archived, because the data might be needed at a later date, such as if the student cannot remember their grades.
  - 7.3. Confirmation for archival must be undergone to ensure the archival is wanted.



- 7.4. Any archived data should be restorable if it is accidentally archived.
- 8. Any information the end user wishes to see must be easily displayed, given they are allowed to see it.
  - 8.1. The user interface – a Windows form – should be clear in terms of its presentation.
  - 8.2. The user interface should be quick and relatively simple to navigate.
  - 8.3. If a user wishes to find out information they do not yet know, they must be able to select what they want to find out from querying a specific item in the database.

## Design

Presenting the solution clearly is just as important as having a solution in the first place. If a solution cannot be presented effectively, and one is not able to understand it, then there may as well be no solution.

Below are various areas of the solution that need planned designs before being taken into account. With these designs, creating the final solution should – in theory – be made easier for me, and have the end product be easily understood by the end user I'm making it for.

## Overall System Design

The overall design encompasses the whole solution into a small and simple table. This sheds a little bit of light onto everything before more detailed explanations of each area are gone through below. Here, though, is everything set at a high level so a summary of the system can be represented cleanly – Be wary that values here may change as the project is developed.

Input	Process	Storage	Output
Searching for Student - ID - Forename - Surname - Tutor group - IT course group Searching for Unit - Unit ID - Unit name Search specific grades	Check database for information on a Student or on a Unit via linked tables.	Database - Stores information on students - Stores information on units	Student information - ID - Forename - Surname - Tutor group - IT course group - Units taken Unit information - Name - Description Students' grades

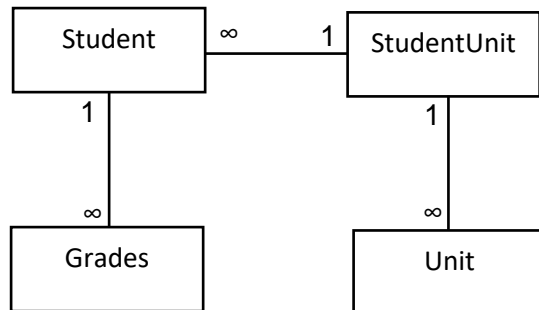
## Entity-Relationship Diagrams

Each of the present values are related to one another in one way or another. Their relation can be visualised by entity-relationship diagrams, designed to show how the information is related. The diagram specifically shows whether data is one-to-one, one-to-many, or many-to-many.

One-to-one relationships are ones which are strictly one value from one entity assigned to one value from another entity.

One-to-many relationships are ones where one value from one entity can have many values from another entity assigned to it. For instance, one student is able to have many grades.

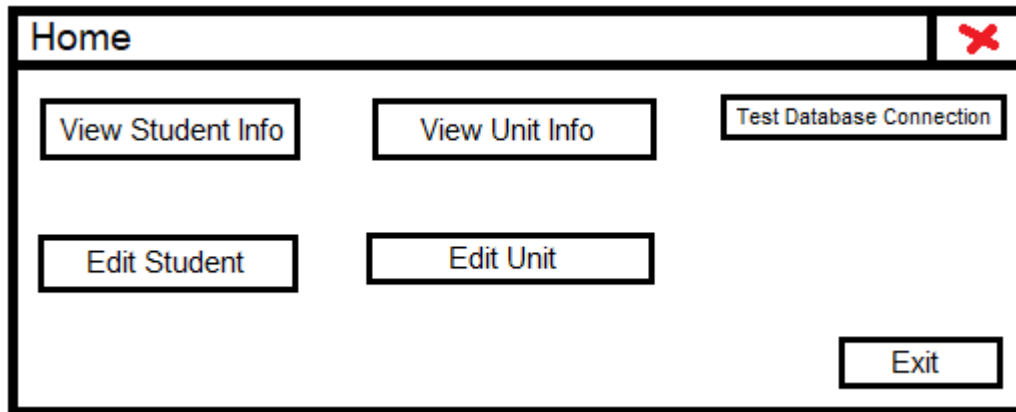
Many-to-many relationships are ones that can have many values of each entity assigned to one another. For instance, one student can be taking many units, and one unit can be taken by many students. Any data that is many-to-many will require a link table between the two for optimal data normalisation.



## User Interface

Of course, the end user needs a simple way to actually access data stored within the database. This is where the user interface comes into play. Having a simple and efficient user interface will allow the end user to view or edit whatever they need.

(1) Home page:

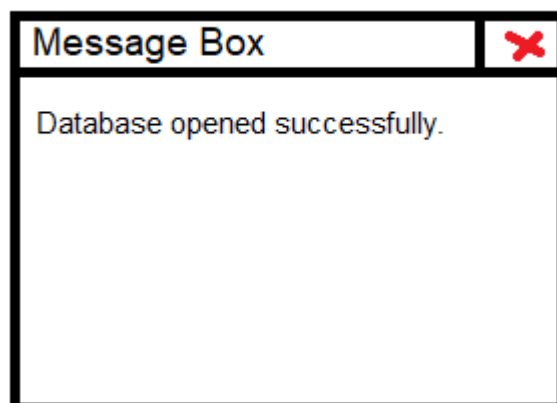


This is where the end user ends up once the database viewing system is loaded initially. They can see various buttons that, when clicked, will take the user to other areas of the database application. From here, they will be able to find their way to each area of the database simply and swiftly.

Pressing the "View Student Info" button will take them to a form where the end user can query as much or as little about a student (or selection of students) as need be. A similar process occurs when the "View Unit Info" button is pressed.

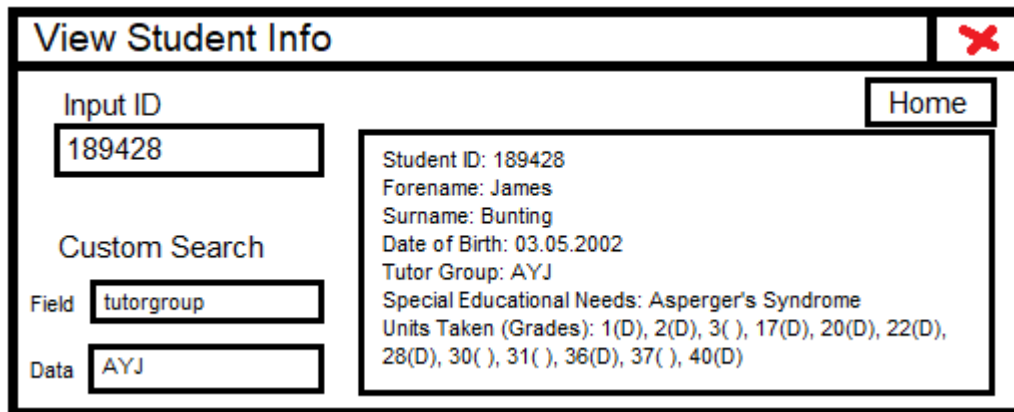
Pressing the "Edit Student" or "Edit Unit" buttons will take the user to their respective edit pages.

Pressing the "Test Database Connection" button will simply test the connection to the database. If an error occurs during the diagnostic, it will be displayed for the user to see. Otherwise, a message box will simply read "Database opened successfully", followed by a message box saying "Database closed successfully", provided the database connection is working.



(2) View Student Info / View Unit Info:

*The example used is "View Student Info". The information described is widely similar between the two, but differences will be described if they arise.*



Both the pages listed above are similar in concept, and shall be explained together. Their purpose is to query the database for information, and then display it for the user.

The “Input ID” area is used to quickly search for a student based on their student ID.

The “Custom Search” area is a bit more complex. The user must input the name of the field in the top box – making sure it’s correct relative to the database declaration. For instance, typing “Tutor Group” would not work. “tutorgroup” has to be typed. For ease of use, a list of the database field names will be displayed by clicking on a help button (which isn’t in the design but will likely be present). Following up, the “Data” area operates much like the “Input ID” field, except the data typed is determined by the field declared in the “Field” area. The big box is where the data is displayed for the user. The fields will be different between the “View Student Info” and “View Unit Info” pages, but they are conceptually the same. The home button is obvious – it sends the user back to the home page, and is present on both the “View Student Info” and the “View Unit Info” pages.

### (3) Edit Student / Edit Unit:

These two pages, much like the viewing equivalents, are both similar but not quite the same. That said, these two are much more alike other than input fields.

## Data Structure

To actually code the database, I would need to write out a few statements using data definition language (DDL). Specifically, I will be using a MySQL database, and below are a few statements to be used within the coded solution:

### Student Database

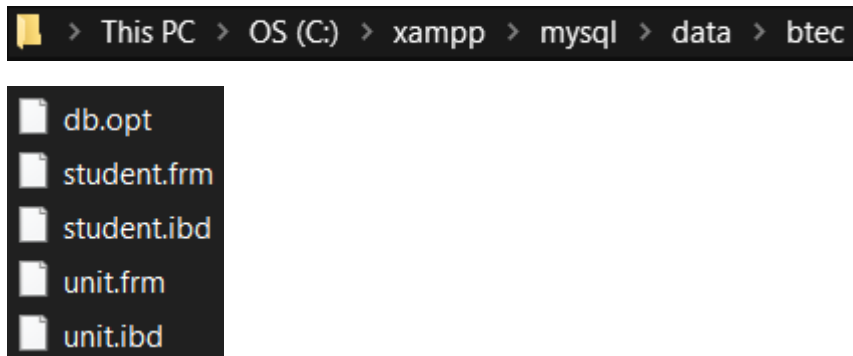
```
Dim StudDDL As String = "CREATE TABLE `btec`.`student` (  
  `studentid` VARCHAR(6) NOT NULL ,  
  `fname` VARCHAR(20) NOT NULL ,  
  `sname` VARCHAR(20) NOT NULL ,  
  `dob` DATE NOT NULL ,  
  `tutorgroup` VARCHAR(3) NOT NULL ,  
  `SEN` TEXT NOT NULL ,  
  PRIMARY KEY (`studentid`)  
);"
```

### Units Database

```
Dim UnitDDL As String = "CREATE TABLE `btec`.`unit` (  
  `unitid` VARCHAR(2) NOT NULL ,  
  `uname` TEXT NOT NULL ,  
  `type` VARCHAR(3) NOT NULL ,  
  PRIMARY KEY (`unitid`)  
);"
```

## Storage

The data storage works in this database by, quite simply, storing everything on the master device. This itself is run through a MySQL plugin called phpMyAdmin which enables access to the data.



Once a link is made between the solution and the data store, the data from the database can be accessed by the solution – and by extension, can be accessed by the end user. From here, they are able to access all the information they may need to know about students, units, the grades achieved, etc.



## Processes

Some theoretical SQL statements, ranging in complexity (explanatory notes will be present):

```
mycommand = New MySqlCommand("SELECT * FROM btec.student WHERE studentid = UserInput")
```

- A very simple statement. It displays every item within the student table that is equal to the end user's input. In this instance, that would be a student's ID, therefore that one student's details are printed out in the output area (though that code is not shown).

```
mycommand = New MySqlCommand("SELECT studentid, sname, fname, unitid, ugrade FROM btec.student, btec.units, btec.grades WHERE unitid = UserInput ORDER BY sname")
```

- This next statement takes every student who's taking a specific unit, and displaying their ID, name, the unit's ID, and their grade on said unit. The query output is also ordered alphabetically by the student's surname.

## Validation

It is inevitable that someone will at any point mess up an input and produce erroneous data. To prevent this, validation will be used to ensure the program does not crash at any point, because of an erroneous input.

Specifically, if the end user is having to type out a field in the dataset, there is every chance they might get it wrong. So the reader doesn't get confused when an erroneous value is inputted, validation will be used on the input variable *before* it's sent into a query.

Field(s)	Validation Checks	Description	Error Message	Data	Caught
studentid	Six characters exactly (e.g. 189428)	The test is done to ensure the student ID, a unique identifier for a student, is a valid six digit number.	"Student ID is not valid – must be six characters long"	9428	Yes
unitid	Two characters exactly (e.g. 03)	To avoid any potential discrepancies when creating the database, unit IDs are always two characters.	"Unit ID is not valid – must be two characters long"	3	Yes
studentid unitid	Numeric characters	This check ensures the input is fully numeric, since all IDs used are all numbers.	"Student ID is not valid – must be a number"	18942A 0E	Yes Yes
fname sname	Twenty characters at most	Though we can't check whether a name is erroneous in terms of characters, we can at least make sure it is the right length.	"Forename is not valid – must be below 20 characters" "Surname is not valid – Must be below 20 characters"	"Abcdefghijkl mnopqrstuvwxyz" "Jonas1"	Yes  No – since the data type is VARCHAR, we cannot check character types easily.

## Testing Strategy

For inputs:

- Validate typical inputs, ones that would be expected to appear.
  - For instance, for a Student ID, something such as 184206 is valid.
- Validate erroneous inputs, ones that would not be expected to appear.
  - For instance, for a Student ID, something such as -100000 is not valid.
- Validate extreme inputs, ones that would not typically be expected, but are still valid.
  - For instance, for a Student ID, something such as 180001 is valid, but not expected.
- The test would be done as a table. For each method of input – the item of the database being tested – the description of what is being tested will explain the input itself, and what would be expected to appear. Following that, a typical, erroneous and extreme data instance are tested, and an outcome for each is shown. An example of the table is seen below.

#	Input	Description	Typical	Expected Outcome	Outcome
			Erroneous		
			Extreme		
1	StudentID	The input of the Student ID into the Students table is tested, making sure that the inputs are valid in terms of length, data type and the data itself.	184206	Valid	[]
			-100000	Invalid	[]
			180001	Valid	[]
2					
3					

For processes:

- Alpha testing
  - For a couple simple and then some more complex SQL/DDDL statements, make sure to test as much about them as possible.
  - For instance, a test involving different statements cases can be set up in a table such as the one below.

#	Process	Description	Expected Outcome	Outcome
1	<SQL statement goes here>	This statement is used to search for a specific student, which in this instance is one written out by the end user – seen in the statement by the placeholder “StudentID”	The student’s details are displayed.	[]
2				[]
3				[]

- Beta testing
  - Create a questionnaire based around the solution, targeted at the end user. Have them work through it and fill out the questions appropriately. See if they can or cannot find anything that needs changing which I did not myself encounter.
  - For instance, a question regarding the user interface would be a good idea, since the end user is able to comment on what they think about the interface. They can say what is good and/or bad about it, giving real useful feedback.
  - Ideally, everything should meet up with the objectives' necessities and/or requirements.

For storage:

- Show the database management system operating working with described screenshots.
  - The screenshots will display the code being run, with explanation, alongside the user interface being displayed at each point via explained screenshots.
  - The performance of this system in accordance with the objectives will be shown, so it can be measured how well it meets the objectives.

## Code Dump

The project itself. Every line of code used in the VB.net project is used here.

### Home.vb

```
Imports MySql.Data.MySqlClient
```

```
Public Class Home
```

```
    Dim mycommand As MySqlCommand
```

```
    Dim myconnection As MySqlConnection = New  
    MySqlConnection("Server=127.0.0.1;Database=btec;Uid=root;Pwd=;")
```

```
    Dim myReader As MySqlDataReader
```

```
    Dim DDLstr As String
```

```
    Dim SQLstr As String
```

```
    Private Sub Home_Load(sender As Object, e As EventArgs) Handles MyBase.Load
```

```
        MsgBox("Only close through the Home page.
```

```
Navigate to the home page by clicking the 'Home' button.
```

```
Close with the 'Close' button.
```

```
Navigation buttons are in the bottom right of the window.")
```

```
    ' The above is a warning box for when the program is loaded.
```

```
    ' It warns the user that closing via the X-buttons won't work.
```

```
End Sub
```

```
Private Sub Connectbtn_Click(sender As Object, e As EventArgs) Handles Connectbtn.Click
```

```
    ' The below statement "myconnection" is used to connect to the database.
```

```
    ' If the database needs to be opened for editing, this function is used.
```

```
    ' Once the desired process is finished, the database is closed up again.
```

```
Try
```

```
    myconnection.Open()
```

```
    MsgBox("Database opened successfully.")
```

```
Catch ex As Exception
```

```
    MsgBox("Database not opened.
```

```
" & ex.Message)
```

```
End Try
```

```
Try
```

```
    myconnection.Close()
```

```
    MsgBox("Database closed successfully.")
```

```
Catch ex As Exception
```

```
    MsgBox("Databased not closed.
```

```
" & ex.Message)
```

```
End Try
```

```
End Sub
```

```
Sub CreateTable()
```

```
    Dim StudDDL As String = "CREATE TABLE `btec`.`student` ( `studentid` VARCHAR(6) NOT NULL ,  
`fname` VARCHAR(20) NOT NULL , `sname` VARCHAR(20) NOT NULL , `dob` DATE NOT NULL ,  
`tutorgroup` VARCHAR(3) NOT NULL , `SEN` TEXT NOT NULL , PRIMARY KEY (`studentid`));"
```

```
    Dim UnitDDL As String = "CREATE TABLE `btec`.`unit` ( `unitid` VARCHAR(2) NOT NULL , `uname`  
TEXT NOT NULL , `type` VARCHAR(3) NOT NULL , PRIMARY KEY (`unitid`));"
```

```
    Dim StudUnitDDL As String = "CREATE TABLE `btec`.`studentunits` ( `studentid` VARCHAR(6)  
NOT NULL , `unitid` VARCHAR(2) NOT NULL , `startDate` DATE NOT NULL , `endDate` DATE NOT  
NULL , PRIMARY KEY (`studentid` , `unitid`));"
```

```
    Dim GradesDDL As String = "CREATE TABLE `btec`.`grades` ( `studentid` VARCHAR(6) NOT NULL ,  
`unitid` VARCHAR(2) NOT NULL , `gradeType` VARCHAR(2) NOT NULL , `grade` VARCHAR(1) NOT  
NULL , PRIMARY KEY (`studentid` , `unitid` , `gradeType`));"
```

```
    Dim CourseDDL As String = "CREATE TABLE `btec`.`course` ( `courseid` VARCHAR(3) NOT NULL ,  
`cname` VARCHAR(20) NOT NULL , `courseAmount` VARCHAR(2) NOT NULL , PRIMARY KEY  
(`courseid`));"
```

```
    Dim CourseStudDDL As String = "CREATE TABLE `btec`.`coursestudents` ( `studentid`  
VARCHAR(6) NOT NULL , `courseid` VARCHAR(3) NOT NULL , `unitsComp` VARCHAR(2) NOT NULL ,  
`courseGrade` VARCHAR(4) , PRIMARY KEY (`studentid` , `courseid`));"
```

```
'Data Definition Language (DDL) for the tables.
```

```
End Sub
```

Private Sub ViewStudInfobtn\_Click(sender As Object, e As EventArgs) Handles ViewStudInfobtn.Click

Dim ViewStudInfopage As New ViewStudInfofrm

ViewStudInfopage.Show()

Hide()

' This button is used to display the "View Student Info" page.

' It also hides the "Home" page so that it doesn't get in the way.

' The alike buttons below this operate similarly to this button,

' just for their respective pages

End Sub

Private Sub EditStudbtn\_Click(sender As Object, e As EventArgs) Handles EditStudbtn.Click

Dim EditStudInfopage As New EditStudInfofrm

EditStudInfopage.Show()

Hide()

End Sub

Private Sub ViewStudUnitsbtn\_Click(sender As Object, e As EventArgs) Handles ViewStudGradebtn.Click

Dim ViewStudUnitspage As New ViewStudUnitsfrm

ViewStudUnitspage.Show()

Hide()

End Sub

Private Sub ViewUnitsbtn\_Click(sender As Object, e As EventArgs) Handles ViewUnitsbtn.Click

Dim ViewUnitspage As New ViewUnitsfrm

ViewUnitspage.Show()

Hide()

End Sub

Private Sub ViewCourseInfobtn\_Click(sender As Object, e As EventArgs) Handles ViewCourseInfobtn.Click

Dim ViewCourseInfopage As New ViewCourseInfofrm

ViewCourseInfopage.Show()

Hide()

End Sub

Private Sub ViewStudCoursebtn\_Click(sender As Object, e As EventArgs) Handles ViewStudCoursebtn.Click

Dim CourseStudentsInfopage As New CourseStudentsInfofrm

CourseStudentsInfopage.Show()

Hide()

End Sub

Private Sub FormClosebtn\_Click(sender As Object, e As EventArgs) Handles FormClosebtn.Click

Close()

' Closes the form, and by extension the program.

End Sub

End Class



## ViewStudInfofrm.vb

```
Imports MySql.Data.MySqlClient
```

```
Public Class ViewStudInfofrm
```

```
    Dim mycommand As MySqlCommand
```

```
    Dim myconnection As MySqlConnection = New  
    MySqlConnection("Server=127.0.0.1;Database=btec;Uid=root;Pwd=";
```

```
    Dim myReader As MySqlDataReader
```

```
    Dim DDLstr As String
```

```
    Dim SQLstr As String
```

```
    Private Sub ViewStudInfofrm_Load(sender As Object, e As EventArgs) Handles MyBase.Load
```

```
    End Sub
```

```
    Private Sub ToHomebtn_Click(sender As Object, e As EventArgs) Handles ToHomebtn.Click
```

```
        Home.Show()
```

```
        Close()
```

```
        ' This closes the "View Student Info" page and re-opens the "Home" page.
```

```
    End Sub
```

```
    Private Sub EnterStudIDbtn_Click(sender As Object, e As EventArgs) Handles EnterStudIDbtn.Click
```

```
        Dim htmlSTR As String
```

```
        htmlSTR = "<table style='width:100%' border=2 align='center'>"
```

```
        htmlSTR &= "<tr> <th>Student ID</th> <th>Forename</th> <th>Surname</th> <th>Date of  
        Birth</th> <th>Tutor Group</th> <th>SEN</th> </tr>"
```

```
        If Len(StudentIDInputtxt.Text) = 6 Then
```

```
            SQLstr = "select * FROM student WHERE studentid = @StudentID"
```

```
            myconnection.Open()
```

```
            mycommand = New MySqlCommand(SQLstr, myconnection)
```

```

mycommand.Parameters.Add(New MySqlParameter("StudentID", StudentIDInputtxt.Text))
myReader = mycommand.ExecuteReader()
myReader.Read()

htmlSTR &= ("<tr><td>" & myReader.GetString("studentid") & "</td><td>" &
myReader.GetString("fname") & "</td><td>" & myReader.GetString("sname") &
"</td><td>" & myReader.GetString("dob") & "</td><td>" &
myReader.GetString("tutorGroup") & "</td><td>" & myReader.GetString("SEN") & "</td></tr>")

myconnection.Close()

htmlSTR &= " </table>"

Display.DocumentText = htmlSTR

Else

MsgBox("Invalid ID input.
Input must be six numeric characters.")

End If

End Sub

```

Private Sub StudentAllViewlbl\_Click(sender As Object, e As EventArgs) Handles StudentAllViewlbl.Click

```

Dim htmlSTR As String

htmlSTR = "<table style='width:100%' border=2 align='center'>"

htmlSTR &= "<tr> <th>Student ID</th> <th>Forename</th> <th>Surname</th> <th>Date of
Birth</th> <th>Tutor Group</th> <th>SEN</th> </tr>"

SQLstr = "select * FROM student ORDER BY studentid"

myconnection.Open()

mycommand = New MySqlCommand(SQLstr, myconnection)

myReader = mycommand.ExecuteReader()

While myReader.Read()

htmlSTR &= ("<tr> <td>" & myReader.GetString("studentid") & "</td><td>" &
myReader.GetString("fname") & "</td><td>" & myReader.GetString("sname") &
"</td><td>" & myReader.GetString("dob") & "</td><td>" &
myReader.GetString("tutorGroup") & "</td><td>" & myReader.GetString("SEN") & "</td> </tr>")

```

End While

myconnection.Close()

htmlSTR &= "</table>"

Display.DocumentText = htmlSTR

End Sub

Private Sub EnterUnitIDbtn\_Click(sender As Object, e As EventArgs) Handles EnterUnitIDbtn.Click

Dim htmlSTR As String

htmlSTR = "<table style='width:100%' border=2 align='center'>"

htmlSTR &= "<tr> <th>Unit ID</th> <th>Student ID</th> <th>Forename</th>  
<th>Surname</th> <th>Date of Birth</th> <th>Tutor Group</th> <th>SEN</th>"

If PassCheck.Checked Or MeritCheck.Checked Or DistCheck.Checked Or FailCheck.Checked Then

htmlSTR &= "<th>Grade Type</th> <th>Grade</th>"

End If

htmlSTR &= "</tr>"

If Len(UnitIDInputtxt.Text) = 2 Then

If PassCheck.Checked Or MeritCheck.Checked Or DistCheck.Checked Or FailCheck.Checked  
Then

SQLstr = "SELECT grades.unitid, grades.studentid, student.fname, student.sname,  
student.dob, student.tutorgroup, student.SEN, grades.gradeType, grades.grade

FROM ((grades INNER JOIN student ON grades.studentid = student.studentid) INNER JOIN unit on  
grades.unitid = unit.unitid)

WHERE grades.unitid = @UnitID"

Else

SQLstr = "SELECT studentunits.unitid, studentunits.studentid, student.fname,  
student.sname, student.dob, student.tutorgroup, student.SEN

FROM ((studentunits INNER JOIN student ON studentunits.studentid = student.studentid) INNER  
JOIN unit on studentunits.unitid = unit.unitid)

WHERE studentunits.unitid = @UnitID"

End If

```
If PassCheck.Checked Or MeritCheck.Checked Or DistCheck.Checked Or FailCheck.Checked  
Then
```

```
    SQLstr &= " AND ("
```

```
    If PassCheck.Checked Then
```

```
        SQLstr &= "grades.grade = 'P'"
```

```
    End If
```

```
    If PassCheck.Checked And MeritCheck.Checked Then
```

```
        SQLstr &= " OR "
```

```
    End If
```

```
    If MeritCheck.Checked Then
```

```
        SQLstr &= "grades.grade = 'M'"
```

```
    End If
```

```
    If (PassCheck.Checked And DistCheck.Checked) Or (MeritCheck.Checked And  
DistCheck.Checked) Then
```

```
        SQLstr &= " OR "
```

```
    End If
```

```
    If DistCheck.Checked Then
```

```
        SQLstr &= "grades.grade = 'D'"
```

```
    End If
```

```
    If (PassCheck.Checked And FailCheck.Checked) Or (MeritCheck.Checked And  
FailCheck.Checked) Or (DistCheck.Checked And FailCheck.Checked) Then
```

```
        SQLstr &= " OR "
```

```
    End If
```

```
    If FailCheck.Checked Then
```

```
        SQLstr &= "grades.grade = 'F'"
```

```
    End If
```

```
    SQLstr &= ")"
```

```
End If
```

```
myconnection.Open()
```

```
mycommand = New MySqlCommand(SQLstr, myconnection)
```

```
mycommand.Parameters.Add(New MySqlParameter("UnitID", UnitIDInputtxt.Text))
```

```
myReader = mycommand.ExecuteReader()
```

```

While myReader.Read()
    htmlSTR &= "<tr><td>" & myReader.GetString("unitid") & "</td><td>" &
myReader.GetString("studentid") & "</td><td>" & myReader.GetString("fname") &
    "</td><td>" & myReader.GetString("sname") & "</td><td>" &
myReader.GetString("dob") & "</td><td>" & myReader.GetString("tutorgroup") & "</td><td>" &
    myReader.GetString("SEN") & "</td>"

    If PassCheck.Checked Or MeritCheck.Checked Or DistCheck.Checked Or FailCheck.Checked
Then
        htmlSTR &= "<td>" & myReader.GetString("gradeType") & "</td><td>" &
myReader.GetString("grade") & "</td>"

        End If

        htmlSTR &= "</tr>"

End While

myconnection.Close()

htmlSTR &= " </table>"

Display.DocumentText = htmlSTR

Else

    MsgBox("Invalid ID input.
Input must be two numeric characters.")

    End If

End Sub

```

```

Private Sub UnitAllViewlbl_Click(sender As Object, e As EventArgs) Handles UnitAllViewlbl.Click
    Dim htmlSTR As String

    htmlSTR = "<table style='width:100%' border=2 align='center'>"

    htmlSTR &= "<tr><th>Unit ID</th><th>Student ID</th><th>Forename</th>
<th>Surname</th><th>Date of Birth</th><th>Tutor Group</th><th>SEN</th>"

    If PassCheck.Checked Or MeritCheck.Checked Or DistCheck.Checked Or FailCheck.Checked Then

        htmlSTR &= "<th>Grade Type</th><th>Grade</th>"

    End If

    htmlSTR &= " </tr>"

    If PassCheck.Checked Or MeritCheck.Checked Or DistCheck.Checked Or FailCheck.Checked Then

```

```
SQLstr = "SELECT grades.unitid, grades.studentid, student.fname, student.sname,
student.dob, student.tutorgroup, student.SEN, grades.gradeType, grades.grade
FROM ((grades INNER JOIN student ON grades.studentid = student.studentid) INNER JOIN unit on
grades.unitid = unit.unitid)
WHERE "
    Else
        SQLstr = "SELECT studentunits.unitid, studentunits.studentid, student.fname, student.sname,
student.dob, student.tutorgroup, student.SEN
FROM ((studentunits INNER JOIN student ON studentunits.studentid = student.studentid) INNER
JOIN unit on studentunits.unitid = unit.unitid)"
    End If
    If PassCheck.Checked Or MeritCheck.Checked Or DistCheck.Checked Or FailCheck.Checked Then
        If PassCheck.Checked Then
            SQLstr &= "grades.grade = 'P'"
        End If
        If PassCheck.Checked And MeritCheck.Checked Then
            SQLstr &= " OR "
        End If
        If MeritCheck.Checked Then
            SQLstr &= "grades.grade = 'M'"
        End If
        If (PassCheck.Checked And DistCheck.Checked) Or (MeritCheck.Checked And
DistCheck.Checked) Then
            SQLstr &= " OR "
        End If
        If DistCheck.Checked Then
            SQLstr &= "grades.grade = 'D'"
        End If
        If (PassCheck.Checked And FailCheck.Checked) Or (MeritCheck.Checked And
FailCheck.Checked) Or (DistCheck.Checked And FailCheck.Checked) Then
            SQLstr &= " OR "
        End If
        If FailCheck.Checked Then
```

```
        SQLstr &= "grades.grade = 'F'"
    End If
End If

myconnection.Open()
mycommand = New MySqlCommand(SQLstr, myconnection)
mycommand.Parameters.Add(New MySqlParameter("UnitID", UnitIDInputtxt.Text))
myReader = mycommand.ExecuteReader()
While myReader.Read()
    htmlSTR &= "<tr> <td>" & myReader.GetString("unitid") & "</td><td>" &
myReader.GetString("studentid") & "</td><td>" & myReader.GetString("fname") &
    "</td><td>" & myReader.GetString("sname") & "</td><td>" & myReader.GetString("dob")
    & "</td><td>" & myReader.GetString("tutorgroup") & "</td><td>" &
        myReader.GetString("SEN") & "</td>"
    If PassCheck.Checked Or MeritCheck.Checked Or DistCheck.Checked Or FailCheck.Checked
    Then
        htmlSTR &= "<td>" & myReader.GetString("gradeType") & "</td><td>" &
myReader.GetString("grade") & "</td>"
    End If
    htmlSTR &= "</tr>"
End While
myconnection.Close()
htmlSTR &= " </table>"
Display.DocumentText = htmlSTR
End Sub

End Class
```

[EditStudInfofrm.vb](#)

```
Imports MySql.Data.MySqlClient
```

```
Public Class EditStudInfofrm
```

```
    Dim mycommand As MySqlCommand
```

```
    Dim myconnection As MySqlConnection = New  
    MySqlConnection("Server=127.0.0.1;Database=btec;Uid=root;Pwd=;")
```

```
    Dim myReader As MySqlDataReader
```

```
    Dim DDLstr As String
```

```
    Dim SQLstr As String
```

```
    Private Sub EditStudInfofrm_Load(sender As Object, e As EventArgs) Handles MyBase.Load
```

```
End Sub
```

```
    Private Sub ToHomebtn_Click(sender As Object, e As EventArgs) Handles ToHomebtn.Click
```

```
        Home.Show()
```

```
        Close()
```

```
        ' This closes the "Edit Student Info" page and re-opens the "Home" page.
```

```
End Sub
```

```
    Private Sub EditStudentbtn_Click(sender As Object, e As EventArgs) Handles EditStudentbtn.Click
```

```
        Try
```

```
            myconnection.Open()
```

```
            mycommand = New MySqlCommand("INSERT INTO student VALUES (@StudentID,  
@Forename, @Surname, @DateOfBirth, @TutorGroup, @SEN)", myconnection)
```

```
            mycommand.Parameters.Add(New MySqlParameter("@StudentID", studentidtxt.Text))
```

```
            mycommand.Parameters.Add(New MySqlParameter("@Forename", fnametxt.Text))
```

```
            mycommand.Parameters.Add(New MySqlParameter("@Surname", snametxt.Text))
```

```
            mycommand.Parameters.Add(New MySqlParameter("@DateOfBirth", dobtxt.Text))
```

```
            mycommand.Parameters.Add(New MySqlParameter("@TutorGroup", tutorgrouptxt.Text))
```

```
            mycommand.Parameters.Add(New MySqlParameter("@SEN", SENTxt.Text))
```



```
        mycommand.ExecuteNonQuery()
        myconnection.Close()
        MsgBox("Data appended successfully.")
    Catch ex As Exception
        MsgBox("Data did not append successfully.
" & ex.Message)
    End Try
End Sub

Private Sub EditStudentUnitsbtn_Click(sender As Object, e As EventArgs) Handles
EditStudentUnitsbtn.Click
    Try
        myconnection.Open()

        mycommand = New MySqlCommand("INSERT INTO studentunit VALUES (@StudentID,
@UnitID, @StartDate, @EndDate)", myconnection)

        mycommand.Parameters.Add(New MySqlParameter("@StudentID",
StudUnitsstudentidtxt.Text))

        mycommand.Parameters.Add(New MySqlParameter("@UnitID", StudUnitsunitidtxt.Text))

        mycommand.Parameters.Add(New MySqlParameter("@StartDate",
StudUnitsstartDatetxt.Text))

        mycommand.Parameters.Add(New MySqlParameter("@EndDate",
StudUnitsendDatetxt.Text))

        mycommand.ExecuteNonQuery()
        myconnection.Close()
        MsgBox("Data appended successfully.")
    Catch ex As Exception
        MsgBox("Data did not append successfully.
" & ex.Message)
    End Try
End Sub

Private Sub EditGradesbtn_Click(sender As Object, e As EventArgs) Handles EditGradesbtn.Click
    Try
```

```
myconnection.Open()

mycommand = New MySqlCommand("INSERT INTO grades VALUES (@StudentID, @UnitID,
@GradeType, @Grade)", myconnection)

mycommand.Parameters.Add(New SqlParameter("@StudentID",
Gradesstudentidtxt.Text))

mycommand.Parameters.Add(New SqlParameter("@UnitID", Gradesunitidtxt.Text))

mycommand.Parameters.Add(New SqlParameter("@GradeType",
Gradesgradetypetxt.Text))

mycommand.Parameters.Add(New SqlParameter("@Grade", Gradesgradetxt.Text))

mycommand.ExecuteNonQuery()

myconnection.Close()

MsgBox("Data appended successfully.")

Catch ex As Exception

MsgBox("Data did not append successfully.
" & ex.Message)

End Try

End Sub
```

```
Private Sub EditCourseStudentsbtn_Click(sender As Object, e As EventArgs) Handles
EditCourseStudentsbtn.Click
```

```
Try

myconnection.Open()

mycommand = New MySqlCommand("INSERT INTO coursestudents VALUES (@StudentID,
@CourseID, @UnitsComp, @CourseGrade)", myconnection)

mycommand.Parameters.Add(New SqlParameter("@StudentID",
CourseStudstudentidtxt.Text))

mycommand.Parameters.Add(New SqlParameter("@CourseID",
CourseStudcourseidtxt.Text))

mycommand.Parameters.Add(New SqlParameter("@UnitsComp",
CourseStudunitsComptxt.Text))

mycommand.Parameters.Add(New SqlParameter("@CourseGrade",
CourseStudcourseGradetxt.Text))

mycommand.ExecuteNonQuery()

myconnection.Close()
```

```
        MsgBox("Data appended successfully.")  
    Catch ex As Exception  
        MsgBox("Data did not append successfully.  
" & ex.Message)  
    End Try  
End Sub  
  
End Class
```

[ViewStudUnitsfrm.vb](#)

```
Imports MySql.Data.MySqlClient
```

```
Public Class ViewStudUnitsfrm
```

```
    Dim mycommand As MySqlCommand
```

```
    Dim myconnection As MySqlConnection = New  
MySqlConnection("Server=127.0.0.1;Database=btec;Uid=root;Pwd=;")
```

```
    Dim myReader As MySqlDataReader
```

```
    Dim DDLstr As String
```

```
    Dim SQLstr As String
```

```
Private Sub ViewStudUnitsfrm_Load(sender As Object, e As EventArgs) Handles MyBase.Load
```

```
End Sub
```

```
Private Sub ToHomebtn_Click(sender As Object, e As EventArgs) Handles ToHomebtn.Click
```

```
    Home.Show()
```

```
    Close()
```

```
    ' This closes the "View Student Info" page and re-opens the "Home" page.
```

```
End Sub
```

```
Private Sub EnterStudIDbtn_Click(sender As Object, e As EventArgs) Handles EnterStudIDbtn.Click
```

```
    Dim htmlSTR As String
```

```
    htmlSTR = "<table style='width:100%' border=2 align='center'>"
```

```
    htmlSTR &= "<tr> <th>Student ID</th> <th>Forename</th> <th>Surname</th> <th>Unit  
ID</th> <th>Grade Type</th> <th>Grade</th> </tr>"
```

```
    If Len(StudentIDInputtxt.Text) = 6 Then
```

```
        SQLstr = "select grades.studentid, student.fname, student.sname, grades.unitid,  
grades.gradeType, grades.grade
```

```
FROM ((grades INNER JOIN student ON grades.studentid = student.studentid) INNER JOIN unit on  
grades.unitid = unit.unitid)
```

```
WHERE grades.studentid = @StudentID
```

```
ORDER BY grades.studentid ASC"
```

```
myconnection.Open()
```

```
mycommand = New MySqlCommand(SQLstr, myconnection)
```

```
mycommand.Parameters.Add(New SqlParameter("StudentID", StudentIDInputtxt.Text))
```

```
myReader = mycommand.ExecuteReader()
```

```
While myReader.Read()
```

```
    htmlSTR &= ("| <td>" & myReader.GetString("studentid") & "</td> <td>" &
myReader.GetString("fname") & "</td> <td>" & myReader.GetString("sname") &
|  |

```

```
    "</td> <td>" & myReader.GetString("unitid") & "</td> <td>" &
myReader.GetString("gradeType") & "</td> <td>" & myReader.GetString("grade") & "</td> </tr>")
```

```
End While
```

```
myconnection.Close()
```

```
htmlSTR &= " </table>"
```

```
Display.DocumentText = htmlSTR
```

```
Else
```

```
    MsgBox("Invalid ID input.
```

```
Input must be six numeric characters.")
```

```
End If
```

```
End Sub
```

```
Private Sub EnterUnitIDbtn_Click(sender As Object, e As EventArgs) Handles EnterUnitIDbtn.Click
```

```
    Dim htmlSTR As String
```

```
    htmlSTR = "<table style='width:100%' border=2 align='center'>"
```

```
    htmlSTR &= "<tr> <th>Student ID</th> <th>Forename</th> <th>Surname</th> <th>Unit
ID</th> <th>Grade Type</th> <th>Grade</th> </tr>"
```

```
    If Len(UnitIDInputtxt.Text) = 2 Then
```

```
        SQLstr = "select grades.studentid, student.fname, student.sname, grades.unitid,
grades.gradeType, grades.grade
```

```
FROM ((grades INNER JOIN student ON grades.studentid = student.studentid) INNER JOIN unit on
grades.unitid = unit.unitid)
```

```
WHERE grades.unitid = @UnitID
ORDER BY grades.studentid ASC"

myconnection.Open()

mycommand = New MySqlCommand(SQLstr, myconnection)

mycommand.Parameters.Add(New MySqlParameter("UnitID", UnitIDInputtxt.Text))

myReader = mycommand.ExecuteReader()

While myReader.Read()

    htmlSTR &= ("| <td>" & myReader.GetString("studentid") & "</td> <td>" &
myReader.GetString("fname") & "</td> <td>" & myReader.GetString("sname") &
    "</td> <td>" & myReader.GetString("unitid") & "</td> <td>" &
myReader.GetString("gradeType") & "</td> <td>" & myReader.GetString("grade") & "</td> </tr>")

End While

myconnection.Close()

htmlSTR &= " </table>"

Display.DocumentText = htmlSTR

Else

    MsgBox("Invalid ID input.
Input must be two numeric characters.")

End If

End Sub

End Class
|  |

```

`CourseStudentsInfofrm.vb``Imports MySql.Data.MySqlClient``Public Class CourseStudentsInfofrm``Dim mycommand As MySqlCommand``Dim myconnection As MySqlConnection = New  
MySqlConnection("Server=127.0.0.1;Database=btec;Uid=root;Pwd=";``Dim myReader As MySqlDataReader``Dim DDLstr As String``Dim SQLstr As String``Private Sub CourseStudentsInfofrm_Load(sender As Object, e As EventArgs) Handles MyBase.Load``End Sub``Private Sub ToHomebtn_Click(sender As Object, e As EventArgs) Handles ToHomebtn.Click``Home.Show()``Close()``' This closes the "View Course Info" page and re-opens the "Home" page.``End Sub``Private Sub EnterCourseIDbtn_Click(sender As Object, e As EventArgs) Handles  
EnterCourseIDbtn.Click``Dim htmlSTR As String``htmlSTR = "<table style='width:100%' border=2 align='center'>"``htmlSTR &= "<tr> <th>Course ID</th> <th>Student ID</th> <th>Units Completed</th>  
<th>Course Grade</th> </tr>"``If Len(CourseIDInputtxt.Text) = 3 Then``SQLstr = "select *``FROM ((coursestudents INNER JOIN student ON coursestudents.studentid = student.studentid)  
INNER JOIN course ON coursestudents.courseid = course.courseid)`

```
WHERE courseid = @CourseID
```

```
ORDER BY coursestudents.courseid"
```

```
myconnection.Open()
```

```
mycommand = New MySqlCommand(SQLstr, myconnection)
```

```
mycommand.Parameters.Add(New MySqlParameter("CourseID", CourseIDInputtxt.Text))
```

```
myReader = mycommand.ExecuteReader()
```

```
While myReader.Read()
```

```
    htmlSTR &= "<tr> <td>" & myReader.GetString("courseid") & "</td><td>" &
myReader.GetString("studentid") & "</td><td>" & myReader.GetString("unitsComp") & "</td><td>"
&
```

```
    myReader.GetString("courseGrade") & "</td>"
```

```
End While
```

```
myconnection.Close()
```

```
    htmlSTR &= " </table>"
```

```
Display.DocumentText = htmlSTR
```

```
Else
```

```
    MsgBox("Invalid ID input.
```

```
Input must be three alphanumeric characters.
```

```
Make sure the course you are searching for exists.")
```

```
End If
```

```
End Sub
```

```
Private Sub CourseAllViewbtn_Click(sender As Object, e As EventArgs) Handles
CourseAllViewbtn.Click
```

```
    Dim htmlSTR As String
```

```
    htmlSTR = "<table style='width:100%' border=2 align='center'>"
```

```
    htmlSTR &= "<tr> <th>Course ID</th> <th>Student ID</th> <th>Units Completed</th>
<th>Course Grade</th> </tr>"
```

```
    SQLstr = "select *
```

```
FROM ((coursestudents INNER JOIN student ON coursestudents.studentid = student.studentid)
INNER JOIN course ON coursestudents.courseid = course.courseid)
```



```
ORDER BY coursestudents.courseid"
```

```

myconnection.Open()

mycommand = New MySqlCommand(SQLstr, myconnection)

mycommand.Parameters.Add(New MySqlParameter("CourseID", CourseIDInputtxt.Text))

myReader = mycommand.ExecuteReader()

While myReader.Read()

    htmlSTR &= "<tr> <td>" & myReader.GetString("courseid") & "</td><td>" &
myReader.GetString("studentid") & "</td><td>" & myReader.GetString("unitsComp") & "</td><td>"
&

    myReader.GetString("courseGrade") & "</td>"

End While

myconnection.Close()

htmlSTR &= " </table>"

Display.DocumentText = htmlSTR

```

```
End Sub
```

```

Private Sub EnterStudIDbtn_Click(sender As Object, e As EventArgs) Handles EnterStudIDbtn.Click

    Dim htmlSTR As String

    htmlSTR = "<table style='width:100%' border=2 align='center'>"

    htmlSTR &= "<tr> <th>Student ID</th> <th>Course ID</th> <th>Units Completed</th>
<th>Course Grade</th> </tr>"

    If Len(StudentIDInputtxt.Text) = 6 Then

        SQLstr = "select *

FROM ((coursestudents INNER JOIN student ON coursestudents.studentid = student.studentid)
INNER JOIN course ON coursestudents.courseid = course.courseid)

WHERE studentid = @StudentID

ORDER BY coursestudents.studentid"

        myconnection.Open()

        mycommand = New MySqlCommand(SQLstr, myconnection)

        mycommand.Parameters.Add(New MySqlParameter("StudentID", StudentIDInputtxt.Text))

```

```

myReader = mycommand.ExecuteReader()

While myReader.Read()

    htmlSTR &= "<tr><td>" & myReader.GetString("studentid") & "</td><td>" &
myReader.GetString("courseid") & "</td><td>" & myReader.GetString("unitsComp") & "</td><td>"
&

myReader.GetString("courseGrade") & "</td>"

End While

myconnection.Close()

htmlSTR &= " </table>"

Display.DocumentText = htmlSTR

Else

    MsgBox("Invalid ID input.
Input must be six alphanumeric characters.")

End If

End Sub

```

```

Private Sub StudentAllViewlbl_Click(sender As Object, e As EventArgs) Handles
StudentAllViewlbl.Click

```

```

    Dim htmlSTR As String

    htmlSTR = "<table style='width:100%' border=2 align='center'>"

    htmlSTR &= "<tr> <th>Student ID</th> <th>Course ID</th> <th>Units Completed</th>
<th>Course Grade</th> </tr>"

    SQLstr = "select *
FROM ((coursestudents INNER JOIN student ON coursestudents.studentid = student.studentid)
INNER JOIN course ON coursestudents.courseid = course.courseid)
ORDER BY coursestudents.studentid"

    myconnection.Open()

    mycommand = New MySqlCommand(SQLstr, myconnection)

    mycommand.Parameters.Add(New MySqlParameter("StudentID", StudentIDInputtxt.Text))

    myReader = mycommand.ExecuteReader()

    While myReader.Read()

```

```
htmlSTR &= "<tr> <td>" & myReader.GetString("studentid") & "</td><td>" &  
myReader.GetString("courseid") & "</td><td>" & myReader.GetString("unitsComp") & "</td><td>"  
&
```

```
myReader.GetString("courseGrade") & "</td>"
```

```
End While
```

```
myconnection.Close()
```

```
htmlSTR &= " </table>"
```

```
Display.DocumentText = htmlSTR
```

```
End Sub
```

```
End Class
```

[ViewUnitsfrm.vb](#)

```
Imports MySql.Data.MySqlClient
```

```
Public Class ViewUnitsfrm
```

```
    Dim mycommand As MySqlCommand
```

```
    Dim myconnection As MySqlConnection = New  
MySqlConnection("Server=127.0.0.1;Database=btec;Uid=root;Pwd=";
```

```
    Dim myReader As MySqlDataReader
```

```
    Dim DDLstr As String
```

```
    Dim SQLstr As String
```

```
Private Sub ViewUnits_Load(sender As Object, e As EventArgs) Handles MyBase.Load
```

```
End Sub
```

```
Private Sub ToHomebtn_Click(sender As Object, e As EventArgs) Handles ToHomebtn.Click
```

```
    Home.Show()
```

```
    Close()
```

```
    ' This closes the "View Units" page and re-opens the "Home" page.
```

```
End Sub
```

```
Private Sub EnterUnitIDbtn_Click(sender As Object, e As EventArgs) Handles EnterUnitIDbtn.Click
```

```
    Dim htmlSTR As String
```

```
    htmlSTR = "<table style='width:100%' border=2 align='center'>"
```

```
    htmlSTR &= "<tr> <th>Unit ID</th> <th>Unit Name</th> <th>Unit Type</th> </tr>"
```

```
If Len(UnitIDInputtxt.Text) = 2 Then
```

```
    SQLstr = "select * FROM unit WHERE unitid = @UnitID"
```

```
    myconnection.Open()
```

```
    mycommand = New MySqlCommand(SQLstr, myconnection)
```

```
    mycommand.Parameters.Add(New MySqlParameter("UnitID", UnitIDInputtxt.Text))
```

```

    myReader = mycommand.ExecuteReader()
    myReader.Read()

    htmlSTR &= ("| <td>" & myReader.GetString("unitid") & "</td><td>" &
myReader.GetString("uname") & "</td><td>" & myReader.GetString("type") & "</td> </tr>")

    myconnection.Close()

    htmlSTR &= " </table>"

    Display.DocumentText = htmlSTR

Else

    MsgBox("Invalid ID input.
Input must be two numeric characters.")

    End If

End Sub

|  |

```

```

Private Sub UnitAllViewlbl_Click(sender As Object, e As EventArgs) Handles UnitAllViewlbl.Click

    Dim htmlSTR As String

    htmlSTR = "<table style='width:100%' border=2 align='center'>"

    htmlSTR &= "<tr> <th>Unit ID</th> <th>Unit Name</th> <th>Unit Type</th> </tr>"

    SQLstr = "select * FROM unit ORDER BY unitid"

    myconnection.Open()

    mycommand = New MySqlCommand(SQLstr, myconnection)

    myReader = mycommand.ExecuteReader()

    While myReader.Read()

        htmlSTR &= ("|<td>" & myReader.GetString("unitid") & "</td><td>" &
myReader.GetString("uname") & "</td><td>" & myReader.GetString("type") & "</td></tr>")

    End While

    myconnection.Close()

    htmlSTR &= " </table>"

    Display.DocumentText = htmlSTR

End Sub

|  |

```

```
Private Sub EnterStudIDbtn_Click(sender As Object, e As EventArgs) Handles EnterStudIDbtn.Click
```

```
    Dim htmlSTR As String
```

```
    htmlSTR = "<table style='width:100%' border=2 align='center'>"
```

```
    htmlSTR &= "<tr> <th>Student ID</th> <th>Unit ID</th> <th>Unit Name</th>"
```

```
    If PassCheck.Checked Or MeritCheck.Checked Or DistCheck.Checked Or FailCheck.Checked Then
```

```
        htmlSTR &= "<th>Grade</th>"
```

```
    End If
```

```
    htmlSTR &= " </tr>"
```

```
    If Len(StudentIDInputtxt.Text) = 6 Then
```

```
        If PassCheck.Checked Or MeritCheck.Checked Or DistCheck.Checked Or FailCheck.Checked  
Then
```

```
            SQLstr = "SELECT grades.studentid, grades.unitid, unit.uname, grades.grade
```

```
FROM ((grades INNER JOIN student ON grades.studentid = student.studentid) INNER JOIN unit on  
grades.unitid = unit.unitid)
```

```
WHERE grades.studentid = @StudentID"
```

```
        Else
```

```
            SQLstr = "SELECT grades.studentid, grades.unitid, unit.uname
```

```
FROM ((grades INNER JOIN student ON grades.studentid = student.studentid) INNER JOIN unit on  
grades.unitid = unit.unitid)
```

```
WHERE grades.studentid = @StudentID"
```

```
        End If
```

```
        If PassCheck.Checked Or MeritCheck.Checked Or DistCheck.Checked Or FailCheck.Checked  
Then
```

```
            SQLstr &= " AND ("
```

```
            If PassCheck.Checked Then
```

```
                SQLstr &= "grades.grade = 'P'"
```

```
            End If
```

```
            If PassCheck.Checked And MeritCheck.Checked Then
```

```
                SQLstr &= " OR "
```

```
            End If
```

```
If MeritCheck.Checked Then
    SQLstr &= "grades.grade = 'M'"
End If

If (PassCheck.Checked And DistCheck.Checked) Or (MeritCheck.Checked And
DistCheck.Checked) Then
    SQLstr &= " OR "
End If

If DistCheck.Checked Then
    SQLstr &= "grades.grade = 'D'"
End If

If (PassCheck.Checked And FailCheck.Checked) Or (MeritCheck.Checked And
FailCheck.Checked) Or (DistCheck.Checked And FailCheck.Checked) Then
    SQLstr &= " OR "
End If

If FailCheck.Checked Then
    SQLstr &= "grades.grade = 'F'"
End If

SQLstr &= ")"
End If

myconnection.Open()
mycommand = New MySqlCommand(SQLstr, myconnection)
mycommand.Parameters.Add(New MySqlParameter("StudentID", StudentIDInputtxt.Text))
myReader = mycommand.ExecuteReader()

While myReader.Read()
    htmlSTR &= ("|<td>" & myReader.GetString("studentid") & "</td><td>" &
myReader.GetString("unitid") & "</td><td>" & myReader.GetString("uname") & "</td>")

    If PassCheck.Checked Or MeritCheck.Checked Or DistCheck.Checked Or FailCheck.Checked
Then
        htmlSTR &= "<td>" & myReader.GetString("grade") & "</td>"

    End If

    htmlSTR &= "</tr>"

End While
|  |

```

```

    myconnection.Close()

    htmlSTR &= " </table>"

    Display.DocumentText = htmlSTR

Else

    MsgBox("Invalid ID input.
Input must be six numeric characters.")

    End If

End Sub

Private Sub StudentAllViewlbl_Click(sender As Object, e As EventArgs) Handles
StudentAllViewlbl.Click

    Dim htmlSTR As String

    htmlSTR = "<table style='width:100%' border=2 align='center'>"

    htmlSTR &= "<tr> <th>Student ID</th> <th>Unit ID</th> <th>Unit Name</th>"

    If PassCheck.Checked Or MeritCheck.Checked Or DistCheck.Checked Or FailCheck.Checked Then

        htmlSTR &= "<th>Grade Type</th> <th>Grade</th>"

    End If

    htmlSTR &= " </tr>"

    If PassCheck.Checked Or MeritCheck.Checked Or DistCheck.Checked Or FailCheck.Checked Then

        SQLstr = "SELECT grades.studentid, grades.unitid, unit.uname, grades.gradeType,
grades.grade
FROM ((grades INNER JOIN student ON grades.studentid = student.studentid) INNER JOIN unit on
grades.unitid = unit.unitid)
WHERE "

    Else

        SQLstr = "SELECT grades.studentid, grades.unitid, unit.uname
FROM ((grades INNER JOIN student ON grades.studentid = student.studentid) INNER JOIN unit on
grades.unitid = unit.unitid)"

    End If

    If PassCheck.Checked Or MeritCheck.Checked Or DistCheck.Checked Or FailCheck.Checked Then

        If PassCheck.Checked Then

```



```
        SQLstr &= "grades.grade = 'P'"
    End If

    If PassCheck.Checked And MeritCheck.Checked Then
        SQLstr &= " OR "
    End If

    If MeritCheck.Checked Then
        SQLstr &= "grades.grade = 'M'"
    End If

    If (PassCheck.Checked And DistCheck.Checked) Or (MeritCheck.Checked And
    DistCheck.Checked) Then
        SQLstr &= " OR "
    End If

    If DistCheck.Checked Then
        SQLstr &= "grades.grade = 'D'"
    End If

    If (PassCheck.Checked And FailCheck.Checked) Or (MeritCheck.Checked And
    FailCheck.Checked) Or (DistCheck.Checked And FailCheck.Checked) Then
        SQLstr &= " OR "
    End If

    If FailCheck.Checked Then
        SQLstr &= "grades.grade = 'F'"
    End If
End If

myconnection.Open()
mycommand = New MySqlCommand(SQLstr, myconnection)
myReader = mycommand.ExecuteReader()
While myReader.Read()
    htmlSTR &= "<tr> <td>" & myReader.GetString("studentid") & "</td><td>" &
myReader.GetString("unitid") & "</td><td>" & myReader.GetString("uname") & "</td>"
    If PassCheck.Checked Or MeritCheck.Checked Or DistCheck.Checked Or FailCheck.Checked
Then
```

```
        htmlSTR &= "<td>" & myReader.GetString("gradeType") & "</td><td>" &  
myReader.GetString("grade") & "</td>"
```

```
    End If
```

```
    htmlSTR &= "</tr>"
```

```
End While
```

```
myconnection.Close()
```

```
htmlSTR &= " </table>"
```

```
Display.DocumentText = htmlSTR
```

```
End Sub
```

```
End Class
```

[ViewCourseInfofrm.vb](#)

```
Imports MySql.Data.MySqlClient
```

```
Public Class ViewCourseInfofrm
```

```
    Dim mycommand As MySqlCommand
```

```
    Dim myconnection As MySqlConnection = New  
    MySqlConnection("Server=127.0.0.1;Database=btec;Uid=root;Pwd=";
```

```
    Dim myReader As MySqlDataReader
```

```
    Dim DDLstr As String
```

```
    Dim SQLstr As String
```

```
    Private Sub ViewCourseInfofrm_Load(sender As Object, e As EventArgs) Handles MyBase.Load
```

```
End Sub
```

```
    Private Sub ToHomebtn_Click(sender As Object, e As EventArgs) Handles ToHomebtn.Click
```

```
        Home.Show()
```

```
        Close()
```

```
        ' This closes the "View Course Info" page and re-opens the "Home" page.
```

```
End Sub
```

```
    Private Sub EnterCourseIDbtn_Click(sender As Object, e As EventArgs) Handles  
    EnterCourseIDbtn.Click
```

```
        Dim htmlSTR As String
```

```
        htmlSTR = "<table style='width:100%' border=2 align='center'>"
```

```
        htmlSTR &= "<tr> <th>Course ID</th> <th>Course Name</th> <th>Amount of Units</th> </tr>"
```

```
        If Len(CourseIDInputtxt.Text) = 3 Then
```

```
            SQLstr = "select * FROM course WHERE courseid = @CourseID"
```

```
            myconnection.Open()
```

```
            mycommand = New MySqlCommand(SQLstr, myconnection)
```

```

mycommand.Parameters.Add(New MySqlParameter("CourseID", CourseIDInputtxt.Text))
myReader = mycommand.ExecuteReader()
myReader.Read()

htmlSTR &= ("| <td>" & myReader.GetString("courseid") & "</td><td>" &
myReader.GetString("cname") & "</td><td>" & myReader.GetString("courseAmount") & "</td>
</tr>")

myconnection.Close()

htmlSTR &= " </table>"

Display.DocumentText = htmlSTR

Else

MsgBox("Invalid ID input.
Input must be three alphanumeric characters.
Make sure the course you are searching for exists.")

End If

End Sub

Private Sub CourseAllViewbtn_Click(sender As Object, e As EventArgs) Handles
CourseAllViewbtn.Click

Dim htmlSTR As String

htmlSTR = "<table style='width:100%' border=2 align='center'>"

htmlSTR &= "<tr> <th>Course ID</th> <th>Course Name</th> <th>Unit Amount</th> </tr>"

SQLstr = "SELECT * FROM course ORDER BY courseAmount"

myconnection.Open()

mycommand = New MySqlCommand(SQLstr, myconnection)

myReader = mycommand.ExecuteReader()

While myReader.Read()

htmlSTR &= ("|<td>" & myReader.GetString("courseid") & "</td><td>" &
myReader.GetString("cname") & "</td><td>" & myReader.GetString("courseAmount") &
"</td></tr>")

End While

myconnection.Close()

|  |

|  |

```

```
htmlSTR &= " </table>"
```

```
Display.DocumentText = htmlSTR
```

```
End Sub
```

```
Private Sub EnterStudIDbtn_Click(sender As Object, e As EventArgs) Handles EnterStudIDbtn.Click
```

```
Dim htmlSTR As String
```

```
htmlSTR = "<table style='width:100%' border=2 align='center'>"
```

```
htmlSTR &= "<tr> <th>Student ID</th> <th>Course ID</th> <th>Course Name</th> <th>Unit  
Amount</th> <th>Units Completed</th> "
```

```
If GradeCheck.Checked Then
```

```
htmlSTR &= "<th>Grade</th>"
```

```
End If
```

```
htmlSTR &= " </tr>"
```

```
SQLstr = "SELECT coursestudents.studentid, coursestudents.courseid, course.cname,  
course.courseAmount, coursestudents.unitsComp"
```

```
If GradeCheck.Checked Then
```

```
SQLstr &= ", coursestudents.courseGrade "
```

```
End If
```

```
SQLstr &= "
```

```
FROM ((coursestudents INNER JOIN student ON coursestudents.studentid = student.studentid)  
INNER JOIN course ON coursestudents.courseid = course.courseid)
```

```
WHERE coursestudents.studentid = @StudentID"
```

```
If GradeCheck.Checked And Not FailCheck.Checked Then
```

```
SQLstr &= " AND NOT coursestudents.courseGrade = 'F'"
```

```
End If
```

```
SQLstr &= " ORDER BY coursestudents.studentid"
```

```
If Len(StudentIDInputtxt.Text) = 6 Then
```

```
myconnection.Open()
```

```
mycommand = New MySqlCommand(SQLstr, myconnection)
```

```

mycommand.Parameters.Add(New MySqlParameter("StudentID", StudentIDInputtxt.Text))
myReader = mycommand.ExecuteReader()

While myReader.Read()

    htmlSTR &= "<tr><td>" & myReader.GetString("studentid") & "</td><td>" &
myReader.GetString("courseid") & "</td><td>" & myReader.GetString("cname") & "</td><td>" &
    myReader.GetString("courseAmount") & "</td><td>" & myReader.GetString("unitsComp") &
"</td>"

    If GradeCheck.Checked Then

        htmlSTR &= "<td>" & myReader.GetString("gradeType") & "</td>"

    End If

    htmlSTR &= "</tr>"

End While

myconnection.Close()

htmlSTR &= " </table>"

Display.DocumentText = htmlSTR

Else

    MsgBox("Invalid ID input.
Input must be six alphanumeric characters.")

End If

End Sub

```

```

Private Sub StudentAllViewlbl_Click(sender As Object, e As EventArgs) Handles
StudentAllViewlbl.Click

    Dim htmlSTR As String

    htmlSTR = "<table style='width:100%' border=2 align='center'>"

    htmlSTR &= "<tr><th>Student ID</th><th>Course ID</th><th>Course Name</th><th>Unit
Amount</th><th>Units Completed</th> "

    If GradeCheck.Checked Then

        htmlSTR &= "<th>Grade</th>"

    End If

    htmlSTR &= " </tr>"

```

```
SQLstr = "SELECT coursestudents.studentid, coursestudents.courseid, course.cname,  
course.courseAmount, coursestudents.unitsComp"  
  
If GradeCheck.Checked Then  
    SQLstr &= ", coursestudents.courseGrade "  
  
End If  
  
SQLstr &= "  
  
FROM ((coursestudents INNER JOIN student ON coursestudents.studentid = student.studentid)  
INNER JOIN course ON coursestudents.courseid = course.courseid)"  
  
If GradeCheck.Checked And Not FailCheck.Checked Then  
    SQLstr &= " AND NOT coursestudents.courseGrade = 'F'"  
  
End If  
  
SQLstr &= " ORDER BY coursestudents.studentid"  
  
  
myconnection.Open()  
  
mycommand = New MySqlCommand(SQLstr, myconnection)  
  
mycommand.Parameters.Add(New MySqlParameter("StudentID", StudentIDInputtxt.Text))  
  
myReader = mycommand.ExecuteReader()  
  
While myReader.Read()  
  
    htmlISTR &= "<tr> <td>" & myReader.GetString("studentid") & "</td><td>" &  
myReader.GetString("courseid") & "</td><td>" & myReader.GetString("cname") & "</td><td>" &  
myReader.GetString("courseAmount") & "</td><td>" & myReader.GetString("unitsComp") &  
"</td>"  
  
    If GradeCheck.Checked Then  
        htmlISTR &= "<td>" & myReader.GetString("gradeType") & "</td>"  
  
    End If  
  
    htmlISTR &= "</tr>"  
  
End While  
  
myconnection.Close()  
  
htmlISTR &= " </table>"  
  
Display.DocumentText = htmlISTR
```

End Sub

Private Sub FailCheck\_CheckedChanged(sender As Object, e As EventArgs) Handles  
FailCheck.CheckedChanged

    If Not GradeCheck.Checked Then

        FailCheck.AutoCheck = False

    End If

    If GradeCheck.Checked Then

        FailCheck.AutoCheck = True

    End If

    ' Checks whether the Include Grade check box is checked or not.

    ' If not checked, disables use of Include Fails check box.

End Sub

End Class



## Technical Solution Details & Testing

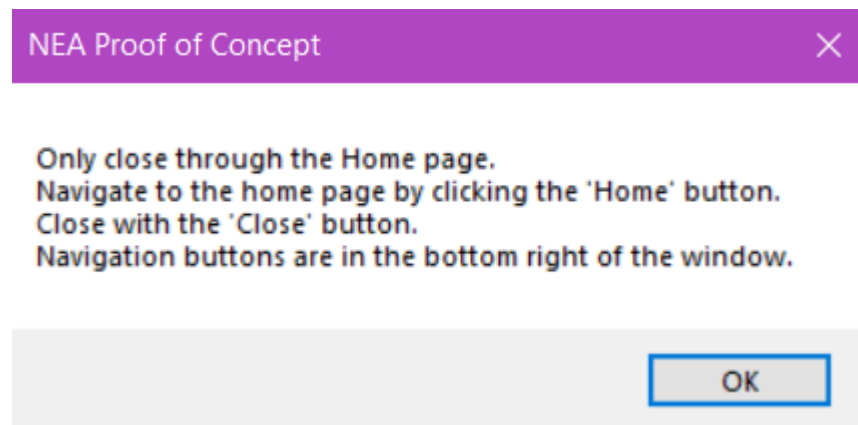
To ensure the project works – and works well – testing needs to be done on each important area, so that no problems would occur when the

[Home.vb](#)

The hub of the happenings – where each other page is linked together, and also where to close the program from.

### Starting pop-up box

When the code is loaded up for the first time, it informs/reminds the end user that using the X buttons is advised against, and should instead be using the Close buttons. Just a simple message box that pops up when the home page is loaded.



```
Private Sub Home_Load(sender As Object, e As EventArgs) Handles MyBase.Load
```

```
    MsgBox("Only close through the Home page.
```

```
    Navigate to the home page by clicking the 'Home' button.
```

```
    Close with the 'Close' button.
```

```
    Navigation buttons are in the bottom right of the window.")
```

```
    ' The above is a warning box for when the program is loaded.
```

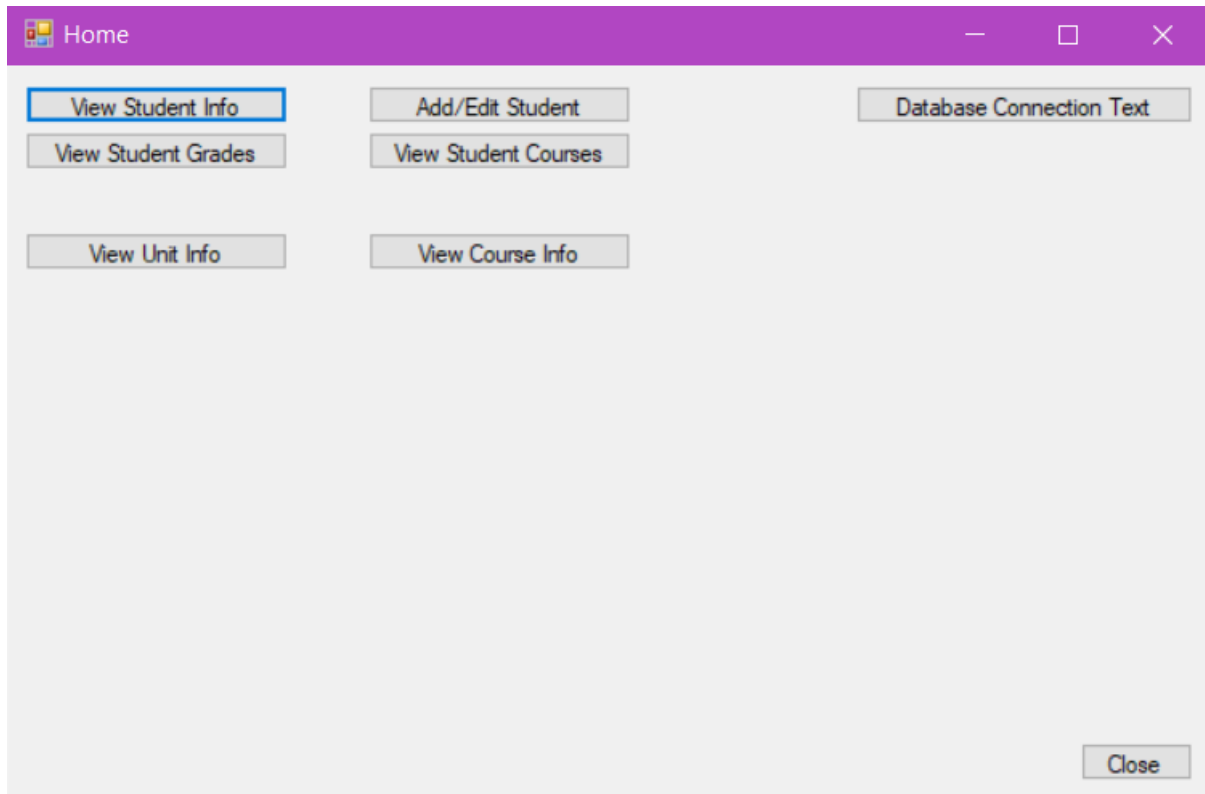
```
    ' It warns the user that closing via the X-buttons won't work.
```

```
End Sub
```

### Home menu overview & database connection test

Once the above text box is closed, the end user is taken to the home menu display. From here, they are able to access every other page within the application, as well as close the application once they are done.

On top of this, the button to the top-right also checks the connection between the application and the database.



Private Sub Connectbtn\_Click(sender As Object, e As EventArgs) Handles Connectbtn.Click

' The below statement "myconnection" is used to connect to the database.

' If the database needs to be opened for editing, this function is used.

' Once the desired process is finished, the database is closed up again.

Try

    myconnection.Open()

    MsgBox("Database opened successfully.")

Catch ex As Exception

    MsgBox("Database not opened.

    " & ex.Message)

End Try

Try

    myconnection.Close()

```
        MsgBox("Database closed successfully.")  
Catch ex As Exception  
        MsgBox("Database not closed.  
" & ex.Message)  
End Try  
  
End Sub
```

### Navigation and Closing

The navigation of the pages is simply by pressing a button and letting it take you to a new page. Once that is done, the

```
Private Sub ViewStudInfobtn_Click(sender As Object, e As EventArgs) Handles  
ViewStudInfobtn.Click
```

```
Dim ViewStudInfopage As New ViewStudInfofrm
```

```
ViewStudInfopage.Show()
```

```
Hide()
```

```
' This button is used to display the "View Student Info" page.
```

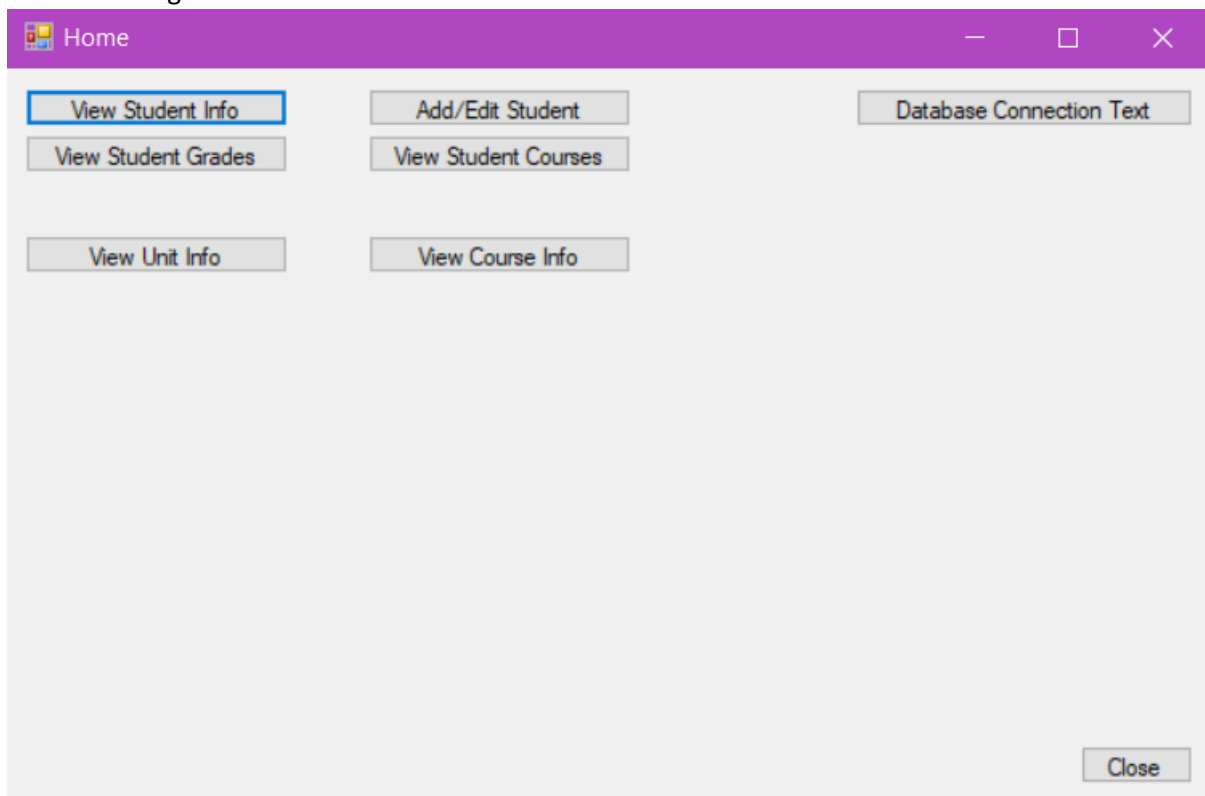
```
' It also hides the "Home" page so that it doesn't get in the way.
```

```
' The alike buttons below this operate similarly to this button,
```

```
' just for their respective pages
```

```
End Sub
```

Before clicking **View Student Info**:



After clicking **View Student Info:**

The screenshot shows a window titled "ViewStudInfofrm" with a purple header bar. Below the header, there are two search sections. The first section, "Search by ID", contains a text input field, an "Enter" button, and a "View All" button. The second section, "Search by Unit", contains a text input field, an "Enter" button, a "View All" button, and a set of checkboxes for "Include Grade" with options "Pass", "Merit", "Distinction", and "Fail". A large empty white area occupies the center of the window. In the bottom right corner, there is a "Home" button.

## ViewStudInfofrm.vb

This page's main purpose is to view information on students

## Search by ID

The Search by ID area is used to either find a specific student by their ID, or to simply view all students. This section is validated by an If statement check which, if an invalid input is input, will flag up with a message box. Otherwise, a query that gathers all the data on that one student is collected.

```
Private Sub EnterStudIDbtn_Click(sender As Object, e As EventArgs) Handles EnterStudIDbtn.Click
```

```
    Dim htmlSTR As String
```

```
    htmlSTR = "<table style='width:100%' border=2 align='center'>"
```

```
    htmlSTR &= "<tr> <th>Student ID</th> <th>Forename</th> <th>Surname</th> <th>Date of Birth</th> <th>Tutor Group</th> <th>SEN</th> </tr>"
```

```
    If Len(StudentIDInputtxt.Text) = 6 Then
```

```
        SQLstr = "select * FROM student WHERE studentid = @StudentID"
```

```
        myconnection.Open()
```

```
        mycommand = New MySqlCommand(SQLstr, myconnection)
```

```
        mycommand.Parameters.Add(New MySqlParameter("StudentID", StudentIDInputtxt.Text))
```

```
        myReader = mycommand.ExecuteReader()
```

```
        myReader.Read()
```



```

        htmlSTR &= ("|<td>" & myReader.GetString("studentid") & "</td><td>" &
myReader.GetString("fname") & "</td><td>" & myReader.GetString("sname") &
        "</td><td>" & myReader.GetString("dob") & "</td><td>" &
myReader.GetString("tutorGroup") & "</td><td>" & myReader.GetString("SEN") & "</td></tr>")

        myconnection.Close()

        htmlSTR &= " </table>"

        Display.DocumentText = htmlSTR

    Else

        MsgBox("Invalid ID input.
Input must be six numeric characters.")

    End If

End Sub

Private Sub StudentAllViewlbl_Click(sender As Object, e As EventArgs) Handles
StudentAllViewlbl.Click

    Dim htmlSTR As String

    htmlSTR = "<table style='width:100%' border=2 align='center'>"

    htmlSTR &= "<tr> <th>Student ID</th> <th>Forename</th> <th>Surname</th> <th>Date of
Birth</th> <th>Tutor Group</th> <th>SEN</th> </tr>"

    SQLstr = "select * FROM student ORDER BY studentid"

    myconnection.Open()

    mycommand = New MySqlCommand(SQLstr, myconnection)

    myReader = mycommand.ExecuteReader()

    While myReader.Read()

        htmlSTR &= ("

|  |

```

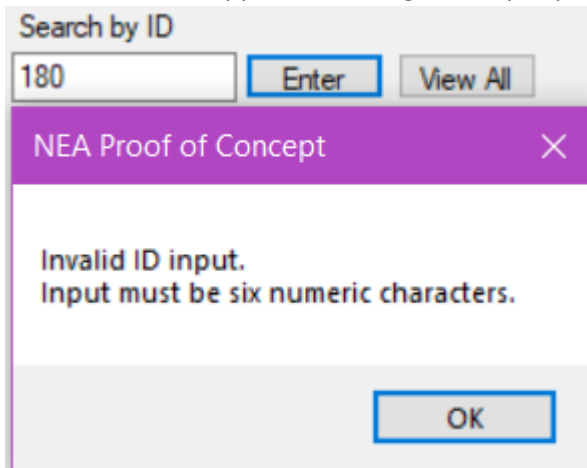
Display.DocumentText = htmlSTR

End Sub

Test 1 – Incorrect input: 180

Expected result, an error box appears and the query does not run.

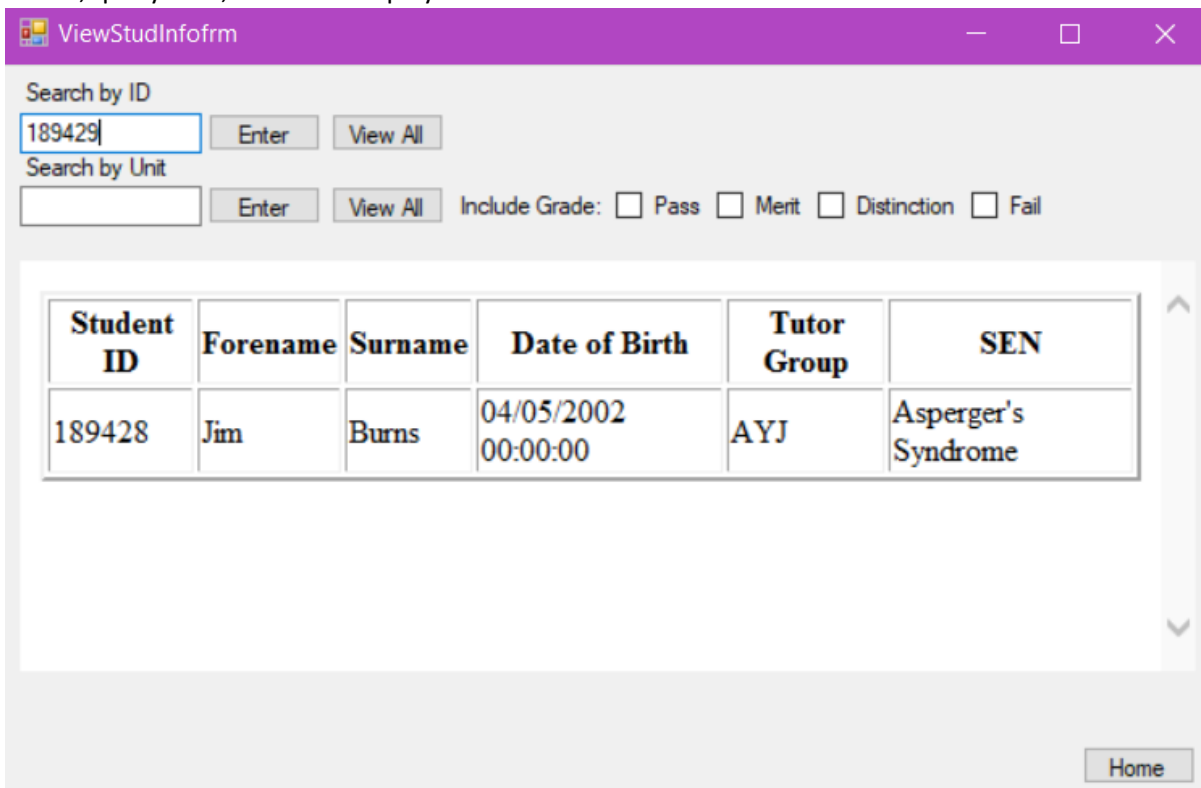
Result, error box appears detailing error, query does not run.



Test 2 – Valid input: 189428

Expected result, query runs and retrieves information on student with ID 189428

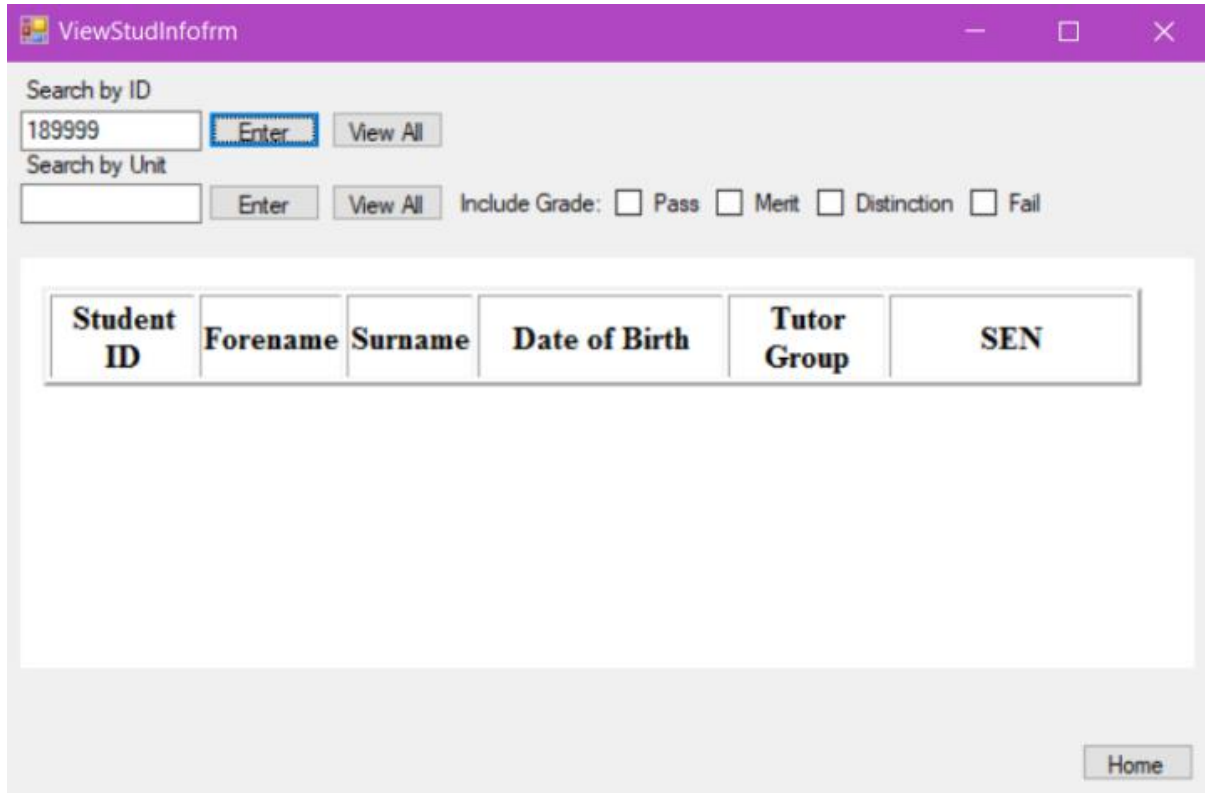
Result, query runs, table box displays data on student with ID 189428



Test 3 – Valid input without a record: 189999

Expected result, query runs and retrieves no information

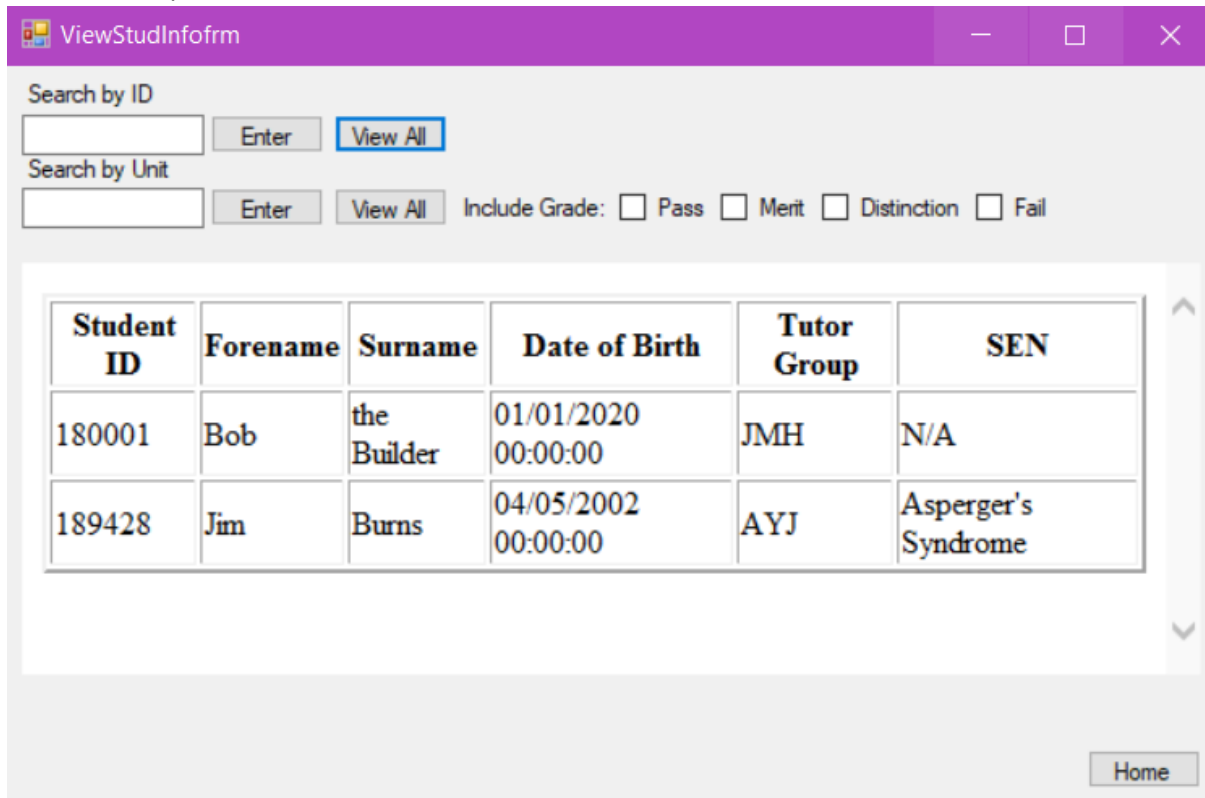
Result, query runs as there is no error flag, but no information is pulled



Test 4 – View all button

Expected result, information on all students (that are logged into the database) is shown

Result, see expected result



### Search by Unit

The Search by Unit area is similar to the Search by ID unit, in that it's searching *for* students. Except this time, the students are searched by and sorted by their units taken. Not only that, but this search can also be narrowed down by achieved grades.

Only code for the View Single button will be displayed, since the SQL query is quite long. The principal is much the same though.

```
Private Sub EnterUnitIDbtn_Click(sender As Object, e As EventArgs) Handles EnterUnitIDbtn.Click
```

```
    Dim htmlSTR As String
```

```
    htmlSTR = "<table style='width:100%' border=2 align='center'>"
```

```
    htmlSTR &= "<tr> <th>Unit ID</th> <th>Student ID</th> <th>Forename</th>
<th>Surname</th> <th>Date of Birth</th> <th>Tutor Group</th> <th>SEN</th>"
```

```
    If PassCheck.Checked Or MeritCheck.Checked Or DistCheck.Checked Or FailCheck.Checked Then
```

```
        htmlSTR &= "<th>Grade Type</th> <th>Grade</th>"
```

```
    End If
```

```
    htmlSTR &= " </tr>"
```

```
    If Len(UnitIDInputtxt.Text) = 2 Then
```

```
        If PassCheck.Checked Or MeritCheck.Checked Or DistCheck.Checked Or FailCheck.Checked
Then
```

```
            SQLstr = "SELECT grades.unitid, grades.studentid, student.fname, student.sname,
student.dob, student.tutorgroup, student.SEN, grades.gradeType, grades.grade
```

```
FROM ((grades INNER JOIN student ON grades.studentid = student.studentid) INNER JOIN unit on
grades.unitid = unit.unitid)
```

```
WHERE grades.unitid = @UnitID"
```

```
        Else
```

```
            SQLstr = "SELECT studentunits.unitid, studentunits.studentid, student.fname,
student.sname, student.dob, student.tutorgroup, student.SEN
```

```
FROM ((studentunits INNER JOIN student ON studentunits.studentid = student.studentid) INNER
JOIN unit on studentunits.unitid = unit.unitid)
```

```
WHERE studentunits.unitid = @UnitID"
```

```
        End If
```

```
        If PassCheck.Checked Or MeritCheck.Checked Or DistCheck.Checked Or FailCheck.Checked
Then
```

```
            SQLstr &= " AND ("
```

```
            If PassCheck.Checked Then
```

```
        SQLstr &= "grades.grade = 'P'"
    End If

    If PassCheck.Checked And MeritCheck.Checked Then

        SQLstr &= " OR "

    End If

    If MeritCheck.Checked Then

        SQLstr &= "grades.grade = 'M'"

    End If

    If (PassCheck.Checked And DistCheck.Checked) Or (MeritCheck.Checked And
DistCheck.Checked) Then

        SQLstr &= " OR "

    End If

    If DistCheck.Checked Then

        SQLstr &= "grades.grade = 'D'"

    End If

    If (PassCheck.Checked And FailCheck.Checked) Or (MeritCheck.Checked And
FailCheck.Checked) Or (DistCheck.Checked And FailCheck.Checked) Then

        SQLstr &= " OR "

    End If

    If FailCheck.Checked Then

        SQLstr &= "grades.grade = 'F'"

    End If

    SQLstr &= ")"

End If

myconnection.Open()
mycommand = New MySqlCommand(SQLstr, myconnection)
mycommand.Parameters.Add(New MySqlParameter("UnitID", UnitIDInputtxt.Text))
myReader = mycommand.ExecuteReader()

While myReader.Read()

    htmlSTR &= "<tr><td>" & myReader.GetString("unitid") & "</td><td>" &
myReader.GetString("studentid") & "</td><td>" & myReader.GetString("fname") &
```

```

"/td><td>" & myReader.GetString("sname") & "</td><td>" &
myReader.GetString("dob") & "</td><td>" & myReader.GetString("tutorgroup") & "</td><td>" &
myReader.GetString("SEN") & "</td>"

```

```

If PassCheck.Checked Or MeritCheck.Checked Or DistCheck.Checked Or FailCheck.Checked
Then

```

```

htmlSTR &= "<td>" & myReader.GetString("gradeType") & "</td><td>" &
myReader.GetString("grade") & "</td>"

```

```

End If

```

```

htmlSTR &= "</tr>"

```

```

End While

```

```

myconnection.Close()

```

```

htmlSTR &= " </table>"

```

```

Display.DocumentText = htmlSTR

```

```

Else

```

```

MsgBox("Invalid ID input.

```

```

Input must be two numeric characters.")

```

```

End If

```

```

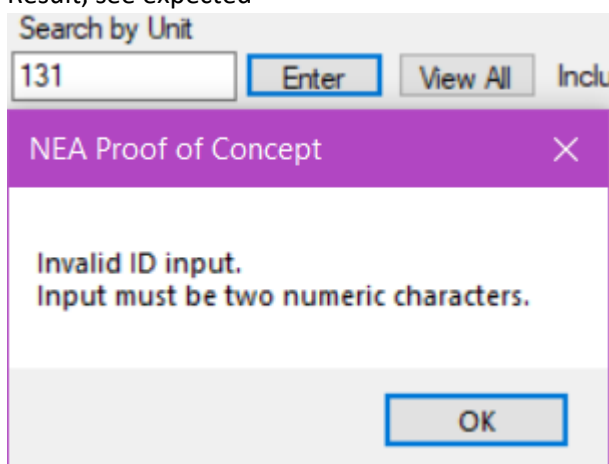
End Sub

```

Test 1 – Enter erroneous input

Expected result, error box flags up, query does not go through

Result, see expected



Test 2 – Enter valid input

Expected result, the information for the students taking that specific unit are displayed

Result, see expected

ViewStudInfofrm

Search by ID  
 Enter View All

Search by Unit  
 Enter View All Include Grade:  Pass  Merit  Distinction  Fail

Unit ID	Student ID	Forename	Surname	Date of Birth	Tutor Group	SEN
02	180001	Bob	the Builder	01/01/2020 00:00:00	JMH	N/A
02	189428	Jim	Burns	04/05/2002 00:00:00	AYJ	Asperger's Syndrome

Home

Test 3 – Enter valid input with fails checked

Expected result, shows all students with a failed unit

Result, see expected

ViewStudInfofrm

Search by ID  
 Enter View All

Search by Unit  
 Enter View All Include Grade:  Pass  Merit  Distinction  Fail

Unit ID	Student ID	Forename	Surname	Date of Birth	Tutor Group	SEN	Grade Type	Grade
02	189428	Jim	Burns	04/05/2002 00:00:00	AYJ	Asperger's Syndrome	INIT	F

Home

Test 4 – View all button, no boxes checked

Expected result, all units currently being taken b student are displayed, alongside the student's data  
Result, see expected

The screenshot shows the 'ViewStudInfofrm' application window. At the top, there are search filters: 'Search by ID' with an empty text box and 'Enter' and 'View All' buttons; 'Search by Unit' with an empty text box, 'Enter' button, and a 'View All' button highlighted in blue. To the right of the 'View All' button are checkboxes for 'Include Grade: Pass', 'Merit', 'Distinction', and 'Fail', all of which are unchecked. Below the filters is a table with the following data:

Unit ID	Student ID	Forename	Surname	Date of Birth	Tutor Group	SEN
01	180001	Bob	the Builder	01/01/2020 00:00:00	JMH	N/A
02	180001	Bob	the Builder	01/01/2020 00:00:00	JMH	N/A
03	180001	Bob	the Builder	01/01/2020 00:00:00	JMH	N/A

A 'Home' button is located at the bottom right of the window.

Test 5 – View all button w/ grades checked

Expected result, shows all *graded* units, excluding fails since they aren't checked

Result, see expected

The screenshot shows the 'ViewStudInfofrm' application window. The search filters are the same as in Test 4, but the 'Include Grade' checkboxes for 'Pass', 'Merit', and 'Distinction' are now checked, while 'Fail' remains unchecked. The 'View All' button is still highlighted in blue. Below the filters is a table with the following data:

Unit ID	Student ID	Forename	Surname	Date of Birth	Tutor Group	SEN	Grade Type	Grade
02	180001	Bob	the Builder	01/01/2020 00:00:00	JMH	N/A	INIT	D
13	180001	Bob	the Builder	01/01/2020 00:00:00	JMH	N/A	INIT	D
02	189428	Jim	Burns	04/05/2002 00:00:00	AYJ	Asperger's Syndrome	RSUB	D

A 'Home' button is located at the bottom right of the window.



### Home button

The home button simply closes this page, and shows the hidden home page. This is the same with each other page, so it will only be covered here.

```
Private Sub ToHomebtn_Click(sender As Object, e As EventArgs) Handles ToHomebtn.Click
```

```
    Home.Show()
```

```
    Close()
```

```
    ' This closes the "View Student Info" page and re-opens the "Home" page.
```

```
End Sub
```

## EditStudInfofrm.vb

The table for adding students to the database, and editing information on a student. All of the areas operate in pretty much the same way, so only Add/Edit Student will be covered in detail.

### Add/edit buttons

The add/edit buttons all operate the same as each other. That is to say, each one uses an INSERT INTO command to append a student into the database.

```
Private Sub EditStudentbtn_Click(sender As Object, e As EventArgs) Handles EditStudentbtn.Click
```

```
Try
```

```
    myconnection.Open()
```

```
    mycommand = New MySqlCommand("INSERT INTO student VALUES (@StudentID, @Forename, @Surname, @DateOfBirth, @TutorGroup, @SEN)", myconnection)
```

```
    mycommand.Parameters.Add(New MySqlParameter("@StudentID", studentidtxt.Text))
```

```
    mycommand.Parameters.Add(New MySqlParameter("@Forename", fnametxt.Text))
```

```
    mycommand.Parameters.Add(New MySqlParameter("@Surname", snametxt.Text))
```

```
    mycommand.Parameters.Add(New MySqlParameter("@DateOfBirth", dobtxt.Text))
```

```
    mycommand.Parameters.Add(New MySqlParameter("@TutorGroup", tutorgrouptxt.Text))
```

```
    mycommand.Parameters.Add(New MySqlParameter("@SEN", SENTxt.Text))
```

```
    mycommand.ExecuteNonQuery()
```

```

myconnection.Close()

MsgBox("Data appended successfully.")

Catch ex As Exception

MsgBox("Data did not append successfully.
" & ex.Message)

End Try

End Sub

```

Test – input Testy Testerson into the database

Expected result, Testy is input into the database, alongside Jim and Bob.

Result,

*Before:*

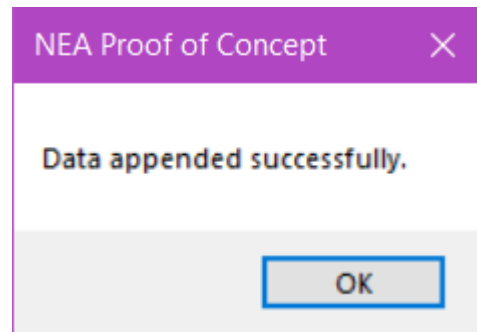
studentid	fname	sname	dob	tutorgroup	SEN
180001	Bob	the Builder	2020-01-01	JMH	N/A
189428	Jim	Burns	2002-05-04	AYJ	Asperger's Syndrome

*During:*

The screenshot shows a web application window titled 'EditStudInfofrm'. It contains several sections for data entry:

- Add/Edit Student:** Fields for Student ID (181111), Forename (Testy), Surname (Testerson), Date of Birth (2010-05-03), Tutor Group (JMH), and SEN (N/A). A 'Database Controls' section with an 'Add/Edit Student' button is on the right.
- Add/Edit Unit taken by Student:** Fields for Student ID, Unit ID, Start Date, and End Date. A button for 'Add/Edit Student's Taken' is on the right.
- Add/Edit Grade of Unit taken by Student:** Fields for Student ID, Unit ID, Grade Type, and Grade. A button for 'Add/Edit Student's Unit's' is on the right.
- Add/Edit Course taken by Student:** Fields for Student ID, Course ID, Complete Units, and Grade. A button for 'Add/Edit Student's Taken' is on the right.

A 'Home' button is located at the bottom right of the window.



After:

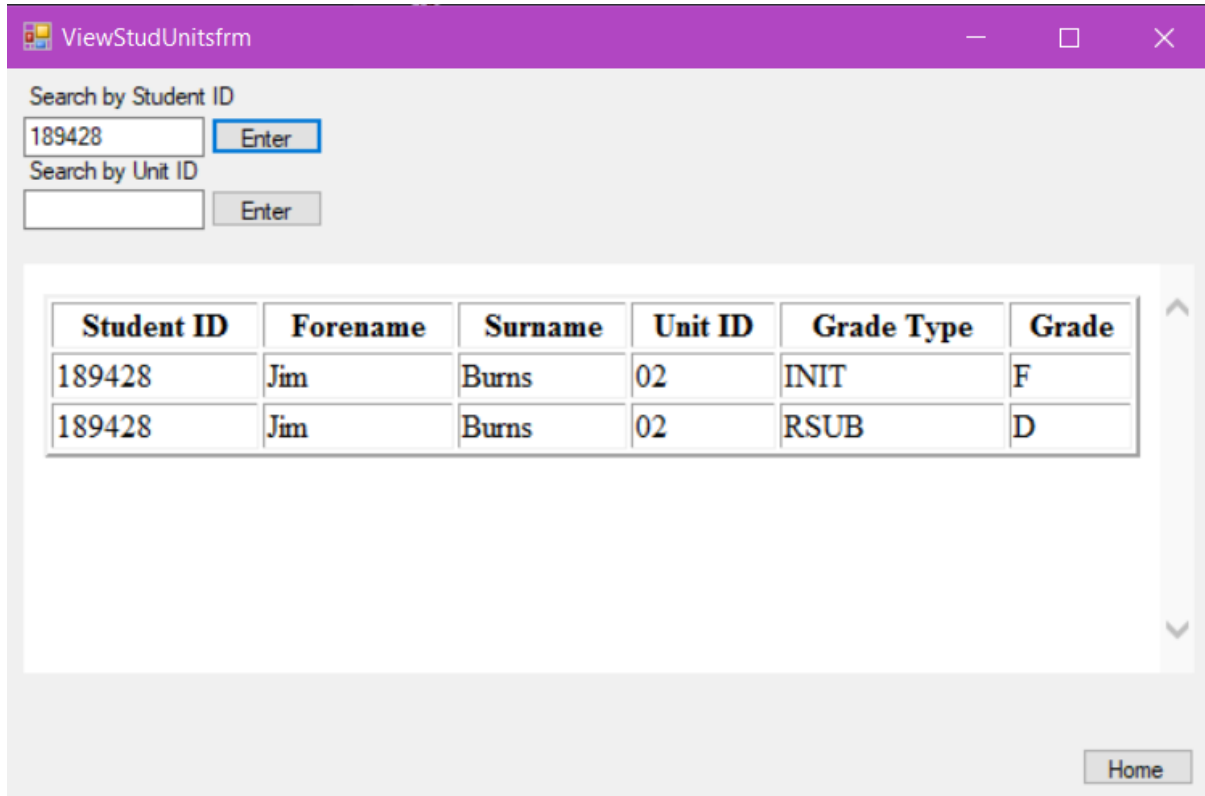
studentid	fname	sname	dob	tutorgroup	SEN
180001	Bob	the Builder	2020-01-01	JMH	N/A
181111	Testy	Testerson	2010-05-03	JMH	N/A
189428	Jim	Burns	2002-05-04	AYJ	Asperger's Syndrome

## Like Pages

The following pages are very similar in design to the `ViewStudentfrm.vb` page. As such, they will not be gone over in as much detail.

### ViewStudUnitsfrm.vb

The grades page takes a single student as an argument, or a single unit as an argument, and searches for each unit taken by said student, or each student taking said unit, respectively.



The screenshot shows a Windows application window titled "ViewStudUnitsfrm". It features two search filters: "Search by Student ID" with a text box containing "189428" and an "Enter" button, and "Search by Unit ID" with an empty text box and an "Enter" button. Below the filters is a table with the following data:

Student ID	Forename	Surname	Unit ID	Grade Type	Grade
189428	Jim	Burns	02	INIT	F
189428	Jim	Burns	02	RSUB	D

A "Home" button is located in the bottom right corner of the window.

ViewStudUnitsfrm

Search by Student ID  
 Enter

Search by Unit ID  
02 Enter

Student ID	Forename	Surname	Unit ID	Grade Type	Grade
180001	Bob	the Builder	02	INIT	D
189428	Jim	Burns	02	RSUB	D
189428	Jim	Burns	02	INIT	F

Home

## CourseStudentsInfofrm.vb

The students' course page displays each student and their taken course (or just one student via the search bar), and displays them in the table as seen below. If searching by course, the results will be alphabetically displayed by the course ID. Otherwise, they will be displayed by the student ID in ascending order.

CourseStudentsInfofrm

Search by Course

Enter View All

Search by Student

Enter View All

Course ID	Student ID	Units Completed	Course Grade
DIP	189428	2	F
SUB	180001	6	D*

Home

CourseStudentsInfofrm

Search by Course

Enter View All

Search by Student

Enter View All

Student ID	Course ID	Units Completed	Course Grade
180001	SUB	6	D*
189428	DIP	2	F

Home

[ViewUnitsfrm.vb](#)

This form is much alike the students info form, in that it displays all the necessary information on a unit, but can also have information linked to a student be displayed.

A student that has taken a unit can be searched for via the Search by Student area, and it can be filtered much like the grades viewer – just with information on the unit being displayed instead of the details of the grade.



ViewUnits

Type Unit ID

Search by Student  
   Include Grade:  Pass  Merit  Distinction  Fail

Unit ID	Unit Name	Unit Type
01	Communication and Employability Skills for IT	CMP
02	Computer Systems	CMP
03	Information Systems	OPT
04	Impact of the Use of IT on Business Systems	OPT
05	Managing Networks	OPT
06	Software Design and Development	OPT
07	...	...

ViewUnits

Type Unit ID

Search by Student  
   Include Grade:  Pass  Merit  Distinction  Fail

Student ID	Unit ID	Unit Name	Grade Type	Grade
180001	02	Computer Systems	INIT	D
180001	13	IT Systems Troubleshooting and Repair	INIT	D
189428	02	Computer Systems	INIT	F
189428	02	Computer Systems	RSUB	D

## ViewcourseInfofrm.vb

The view course info page is used to read up on information about the available courses, as well as students taking each one.

The screenshot shows two instances of the ViewCourseInfofrm application window. The top window displays a list of available courses, and the bottom window displays a list of students taking those courses.

**Top Window: ViewCourseInfofrm**

Type Course ID  
 Enter **View All**

Search by Student  
 Enter **View All**  Include Grade  Include Fails

Course ID	Course Name	Unit Amount
CRT	Certificate	03
SUB	Sub-Diploma	06
90C	90-Credit	09
DIP	Diploma	12
EXT	Extended Diploma	18

Home

**Bottom Window: ViewCourseInfofrm**

Type Course ID  
 Enter **View All**

Search by Student  
 Enter **View All**  Include Grade  Include Fails

Student ID	Course ID	Course Name	Unit Amount	Units Completed
180001	SUB	Sub-Diploma	06	6
189428	DIP	Diploma	12	2

Home

## Evaluation

To conclude with this, the project as a whole will be summed up.

Initially, work on it was okay, if not rather slow. That said, the slowness caught up to me and left me with too little time to do too much. Compromises were made, and a few slight corners were cut, but the project as a whole ended up getting through the proposed requirements pretty well.