

Software Systems Development

Centre Number: 71348

Candidate Number: 0075

Index

Section 1 Planning 	6
Background	7
Problems with Current System	10
Section 2 Analysis 	12
System Requirements	13
Interview	13
Questionnaires.....	14
User Requirements	15
Project Constraints.....	20
Costs.....	21
Staff.....	22
Timescale	23
Legalisation	23
Project plan	27
Methodologies.....	29
Dynamic systems development method (DSDM)	29
Rapid Application Development (RAD)	31
Waterfall	32
SCRUM	39
Extreme Programming (XP).....	45
Chosen Methodology.....	52
Chosen Focus	52
Section 3 Design 	53
GANTT chart.....	54
ER Diagram.....	54
Normalisation of data	54
What is Normalisation	54
0NF	56
1NF	56
2NF	57
3NF	57
Inputs	58
Data Dictionary	59

Student Table	59
Teacher Table.....	67
Scheduled Lessons Table.....	73
Lesson Purchase Table	78
Lesson Bundle Table.....	82
Room Table	85
Instrument Table.....	87
Grade Table.....	89
Achieved Lessons Table	90
Reallocated Student table.....	91
Processes.....	95
Outputs	96
Draft Story Boards.....	96
1) Splash Screen	96
2) Selection Menu	97
3) Private Tuition Menu	97
4) Students List.....	98
5) Teachers List.....	98
6) Instrument List.....	99
7) Grade List	99
8) Lesson Bundle List.....	100
9) Schedule Calendar [Date Selection].....	100
10) Schedule Calendar [Time Selection]	101
11) Schedule Calendar [Scheduled Lessons]	101
12) Schedule Lessons List	102
13) Admin Login	102
14) Add Field	103
15) Delete Confirmation.....	103
Final Story Boards	104
1) Splash Screen	104
2) Selection Menu	105
3) Private Tuition Menu	106
4) Students List.....	107
5) Teachers List.....	108
6) Instruments List	109
7) Grade List	110

8) Lesson Bundle List.....	111
9) Schedule Calendar [Date Selection].....	112
10) Schedule Calendar [Time Selection]	113
11) Schedule Calendar [Scheduled Lessons]	114
12) Schedule Lessons List	115
13) Purchased Lessons List.....	116
14) Admin login	117
15) Add Field	118
16) Delete Confirmation.....	119
Pseudo code.....	120
Splash Screen	120
Main Menu.....	120
Private Tuition Menu	120
Student Form	121
Teacher Form	122
Instrument Form	122
Grade Form	123
Lesson Bundle Form.....	123
Room Form.....	124
Purchased Lessons Form.....	125
Scheduled Lessons Form.....	126
Schedule Calendar (Dates).....	127
Schedule Calendar (Times).....	127
Schedule Calendar (Lesson Details)	127
Deletion Confirmation	128
Login.....	128
Add New Field Form.....	128
Chapter 4 Testing 	131
Test Plan.....	132
2) Forms	132
Test Evidence	267
2) Forms	267
Chapter 5 Evaluation 	571
System Evaluation	572
System Purpose and Operations.....	572
Meeting the user Requirements.....	576

How has data modelling improved system.....	590
Review of end user feedback.....	590
Future recommendations	591
Personal Performance.....	592
Appendices.....	598
Appendix 1 System Questionnaires 	599
Initial Questionnaires.....	600
Final Questionnaires	606
Appendix 2 GANNT Chart 	609
Appendix 3 ER Diagram 	614
What is an ER Diagram?	615
Diagram Key	615
Current System ER Diagram	616
Explanation of Relationships.....	617
Appendix 4 Final System Overview 	621
Forms	622
Code	635

| Section 1 |

| Planning |

Background

The Mitchell school of music is a newly situated musical school located around the area of Coleraine. It has been stated that the current school building comes equipped with 12 individual practice rooms to provide students with a one-on-one lesson with a teacher, along with 4 larger class rooms, which provide students and teachers with the ability to work as a group. Finally, the school also offers a large hall, which could be used for an orchestra display or training, and also house various events accordingly. Each teacher present within the school is either presented with their own room, or a shared room for different shifts, however either way this means that the students will be assigned a relevant room with ease. The owner has stated a concern for the lack of services around the relevant area which seek to encourage and reinforce various musical skills which students possess, and that this was one of the many things which influenced his decision to organise and establish the school.

In regards to accessibility, it has been stated by the owner, Peter that the school has very few restrictions in relations to the age of their students. This means that practically anybody – with a few exceptions, is able to partake within variety of classes and the teaching of musical instruments. Additionally, it is important to address the fact that the school has been segmented into a number of different categories, including; “The specialist weekend school”, “Private tuition”, and “The specialist summer school”. It is important to identify this point, due to the fact that each category acts independently from one another to some degree, and therefore have different times and dates regarding accessibility.

Private tuition is available from September to June - as these months are occupied by the specialist summer school, and provides students with the ability to schedule lessons throughout the week, however not during the weekend. This is due to the fact that the specialist weekend school runs along during this time. To be more specific, students are provided with the ability to schedule private lessons from 1pm-9pm during Monday – Thursday, and 1pm-8pm during a Friday.

The management present within the Mitchell school of music, Peter Mitchell has made numerous comments in relations to the various teachers present within the school. It has been stated that Peter was able to meet and contract various skilled musicians who has met through his career. It should be noted that all teachers present within the school are professionals, and specialise within a specific instrument group or individualised instrument. In total there are a total of 16 teachers present within the school, along with a janitor, and an IT specialist.

Each of these teachers are provided with an individualised room each of which are designed in a manner to benefit from a specific specialisation. It should be noted however, that there have been various instances where teachers will share rooms accordingly. As mentioned slightly earlier, each of these rooms is provided with access to the necessary and corresponding resources needed to operate. This includes the specific instruments which would be stored and used within the school.

Private tuition

As previously mentioned, the private tuition is simply a one-one-one lesson between a student and a teacher. It is important to note that while scheduling a specific lesson, the student is presented with the ability to select the instrument which they will be learning from a category list of: String, Wood Wind, Brass, Percussion, and vocal. Additionally however, before a student is able to schedule lessons, they must first pay a non-returnable tuition fee of £30. Once this fee has been paid, the student would then complete a short examination and be assigned a grade ranging from; Beginner, Intermediate, Advanced and Diploma accordingly. Before advancing further, it is also worth noting that the staff within the school encourage the students to prepare and participate within grade examinations to eventually improve their current grade. One final note to make in regards to grade is that it will directly affect the total cost for lessons, using the pricing system shown below:

	Grade	Fee/30 min weekly lesson
Tuition Fee (No Discount)		£30
Beginner	Initial, Grade 1, Grade 2	£10
Intermediate	Grade 3, Grade 4, Grade 5	£15
Advanced	Grade 6, Grade 7, Grade 8	£20
Diploma		£25
Concessionary – (over 60 or special circumstances)		20% Discount

Interestingly, to become a student at this school you will need to separately purchased lesson blocks, which come in the form of “10 Lessons”, “20 Lessons”, or “30 Lessons”. Additionally, it should be noted that these lessons can be scheduled in a similar manner to that of school terms. In regards to the scheduling system being used within this school, Peter has mentioned the fact that student have been provided with the ability to begin their lessons whenever they desire, with the only requirement being that there is a position available. It is also important to acknowledge the fact that each student is only allowed 30 minutes’ tuition in a single week, unless they have purchased and used another block of lessons.

In regards to payment, to simplify the process, all students are provided with the ability to purchase these lesson blocks in advance. Additionally, to encourage the purchase of more lesson bundles, purchasing “20 Lessons” will provide a 5% discount, while “30 Lessons” will provide a 10% discount. On this regard, it is important to address and acknowledge the calculations which are used within the organisation to calculate total Lesson cost. This is found by multiplying the number of lessons with the grade fee associated with the student. Then depending on the block of lessons which were purchased, the corresponding discount will be used to subtract the correct value of money.

As previously mentioned, this school uses placements as to prevent overflow, and to effectively manage their students. If applying, it is important to understand that to properly manage the inflow of students and general popularity associated with the school, the decision was made to reallocate students should their attendance fall. This is done to prevent the wastage of time and resources of both the school and the teachers involved.

Problems with Current System

After speaking with Peter and conducting a thorough analyse of the school, we have been able to identify numerous issue associated with the current system. In advance to addressing these issues, it could be deemed beneficial to further analyse the current system in regards to operation and performance. The first important aspect to note is that the system would be identified as "Paper-based". This simply means that that staff and management use physical means for collecting, storing and distributing information throughout the school, and externally to students or teachers. Examples of collection methods commonly come in the form of; paper, files, forms etc. Then in regards to storage, one of the most common examples would be filing cabinet, as this provides a physical space for the documentation.

To this day, this system is regarded as outdated however, as technological advancements and implementation within businesses and other areas have become extremely common and beneficial. There are many simple issues within these systems, for instance to collect information, an employee present within the school must take the time and effort to manually record each detail, then they must proceed to store it within an appropriate location where it can easily be accessed. These issues in particular may seem quite minuscule, however as the school continues to grow in regards to students, purchased lessons and scheduled lessons, it become apparent that there is simply too much documentation to be done individually or simply by hand. Additionally, this could lead to storage and duplication issues, due to the fact that may become difficult to locate specific documents, and multiple documents surrounding the same data subject and information may be recorded, one of which could contain inaccurate information.

A few of the specific issues which were identified within the school were:

- ❖ **Tracking Payments** – It has been noted that tracking the payments from students has recently become difficult, and has even been the cause for various billing issues.
- ❖ **Tracking Lessons** – Due to the increase number of students within the school, and the numerous pieces of data which much be recorded, added, removed and updated accordingly, Peter has noted that keeping track of a student and this teachers schedule has become difficult.
- ❖ **Lesson Timetables** – Currently, Peter has taken on the responsibility of manually planning each teacher's time table. This however has proven to be an extremely difficult and time consuming task, due to the fact that he must identify the time

slot, ensuring that it does not overlap with another lesson, and also ensure that it suits the teacher in question.

- ❖ Data Transfer – Another important job which Peter has taken the responsibility for relates to the transfer of data. During the weekends, he attempts to transfer all collected information, however due to numerous issues, it has been stated that this process is not perfect, meaning that it is extremely likely that some important pieces of data have been transferred inaccurately; otherwise some records have even been left incomplete.
- ❖ Attendance – Another important piece of data which has been difficult to record and track accurately within the system is attendance. There have even been instances where students have been asked to leave, due to poor record accuracy.
- ❖ Examination entries – It has been stated that managing the examination entries for the students has proven to be extremely time consuming. It is also important to understand that the teachers are responsible for these records, and thus this takes time away from teaching or other activities.
- ❖ Duplication – Students are presented with the ability to retake exams, however each time an exam is taken, and the student's data is recorded and stored, effectively duplicating the information.
- ❖ Upholding rules – Peter has established various rules within the school, with a few examples being to not accept late entries, nor send an entry for which payment has not been received. The problem however, is that there have been instances in which both have passed entries, and the money has been lost.
- ❖ Result Management – Once received, the examination results need to be photocopied and filed correctly within the school. In addition to this, these documents must be signed by the student/guardian. There have been various instances however in which these documents have been poorly maintained, with cases where records have not been available, certificates go missing or the rise of various disputes.
- ❖ Profits – The poor management and general handling of student records and examination procedures inevitably lead to an overall loss of profit, and has been identified as the cause of numerous problems with billing. Peter has expressed the overall struggle associated with invoicing and receipts using this system.

| Section 2 |

| Analysis |

System Requirements

Interview

- *[Question 1]* – “May I press a query regarding your position within Mitchell School of Music, and the associated responsibilities?”

“I am the owner and manager of Mitchell’s school of Music, which can be a brief indication of the numerous and complex responsibilities and tasks which fall upon my shoulders. For example, I have personally taken it upon myself to complete regular transfer, modifications and general management of the information collected and stored in relations to our students, teachers, lessons and much more.”

- *[Question 2]* – “I would like to ask your overall opinion on the current information management system.”

“I believe that it is important to address the fact that the traditional system has met the demands imposed by the school, providing the ability to collect, store and process relevant information. Yet I have contacted you with this proposal, for while there has always been some form of limitations present within this system, due to the popularity increase regarding the school and continued operations, many more drastic limitations and harmful consequences have been exposed and identified.”

- *[Question 3]* – “You mentioned the fact that there are various problems and limitations associated with the system, would you care to elaborate with more detail?”

“The predominant issues which have become apparent within the information management system can be contributed to the increase of information collection, processing and storage. This has recently made the process of creating, managing and updating various records present within the school tremendously difficult, and would commonly result in the storage of erroneous information, or loss of information in extreme instances.”

- *[Question 4]* – “A system like this can be very complex, so is there any specific content which you would like implemented?”

“In regards to the actual table structure, I have personally identified a few tables which I would like to see present within the system. These include: Students, Teachers, Purchased Lessons and Scheduled Lessons. I should mention however that these are potential just a few of the most important tables, and I would be open to future suggestions for new additions.

In regards to function, I guess that it would be relatively self-explanatory to express the fact that I would like to see table-based forms present within the system, as to provide the users with a clean and structured way to view table information. On this topic, I would also like to see data modification functions implemented, including Record addition, update, and deletion.

In addition to this, I would also like a few specialised forms that filter out specific information, for instance if you deem it plausible I believe that it would be extensively beneficial and interesting to have a scheduled calendar, which would provide users with the ability to select a specific date, and view associated records.

Finally, I would like to address the fact that there are a few attributes that may need calculated by the system. For instance, the database should detail the total bundle cost associated with each purchased lesson, which is a calculation of the student grade fee, selected lesson bundle fee, and lesson bundle multiplier.”

- *[Question 5]* – “To expand upon this point, would you like to express any preferences regarding the system design?”

“As may be expected, I simply want an organised and relatively professional looking system. In regards to colour schemes I would like the system to represent the school itself, therefore I would appreciate it if you could try and implement an assortment of blues. On the note of colours however, it may be beneficial to generally colour code buttons within the system, as to represent their function.”

Questionnaires

Please Reference Appendix 1.

User Requirements

User requirements could simply be described as a collection of points which the program or object in question should meet, hence they are commonly used as a guideline within the planning and development stage of said program or object. For this program specifically, the user requirements will likely detail the various queries, tables and functions which need to be present, along with additional design choices.

Overall, having communicated with both the employees and management present within the school, and reviewed the current information management system, it was possible to assess and assign the following user requirements:

1) Splash Screen

- A) Must initialise and load database system.
- B) Must provide loading progression graphic(s).
- C) Must provide text-based user feedback.
- D) Must link to Main Menu form.
- E) This must be the initial form to load.

2) Main Menu form

- A) Must provide a link to the private tuition Form.
- B) Must display potential links (Unused buttons) to other sections of the database.
- C) Must provide ability to exit program.

3) Private Tuition Form

- A) Must provide a link to Student Table.
- B) Must provide a link to Teacher Table.
- C) Must provide a link to Instrument Table.
- D) Must provide a link to Room Table.
- E) Must provide a link to Grade Table.
- F) Must provide a link to Lesson Bundle Table.
- G) Must provide a link to Scheduled Lessons Table.
- H) Must provide a link to Purchased Lessons Table.
- I) Must provide a link to Schedule calendar.

- J) Must provide a link to Login.
- K) Must provide a link back to Main Menu.
- L) Must present feedback regarding login status.

4) Student Table

- A) Must display student information in an ordered and easy to understand fashion.
- B) Must be able to easily navigate between student records.
- C) Must be able to search for students.
- D) Must provide link to Add field form.
- E) Must provide ability to append details.
- F) Must provide a link to Login or Confirmation form depending if no user is logged in.
- G) Must provide a link back to Private tuition.

5) Teacher Table

- A) Must display teacher information in an ordered and easy to understand fashion.
- B) Must be able to easily navigate between teacher records.
- C) Must be able to search for teachers.
- D) Must provide link to Add field form.
- E) Must provide ability to append details.
- F) Must provide a link to Login or Confirmation form depending if no user is logged in.
- G) Must provide a link back to Private tuition.

6) Instrument Table

- A) Must display instrument information in an ordered and easy to understand fashion.
- B) Must be able to easily navigate between instrument records.
- C) Must be able to search for instruments.
- D) Must provide link to Add field form.
- E) Must provide ability to append details.

- F) Must provide a link to Login or Confirmation form depending if no user is logged in.
- G) Must provide a link back to Private tuition.

7) Room Table

- A) Must display Room information in an ordered and easy to understand fashion.
- B) Must be able to easily navigate between Room records.
- C) Must be able to search for Rooms.
- D) Must provide link to Add field form.
- E) Must provide ability to append details.
- F) Must provide a link to Login or Confirmation form depending if no user is logged in.
- G) Must provide a link back to Private tuition.

8) Grade Table

- A) Must display Grade information in an ordered and easy to understand fashion.
- B) Must be able to easily navigate between Grade records.
- C) Must be able to search for Grades.
- D) Must provide link to Add field form.
- E) Must provide ability to append details.
- F) Must provide a link to Login or Confirmation form depending if no user is logged in.
- G) Must provide a link back to Private tuition.

9) Lesson Bundle Table

- A) Must display Lesson Bundle information in an ordered and easy to understand fashion.
- B) Must be able to easily navigate between Lesson Bundle records.
- C) Must be able to search for Lesson Bundle.
- D) Must provide link to Add field form.
- E) Must provide ability to append details.

- F) Must provide a link to Login or Confirmation form depending if no user is logged in.
- G) Must provide a link back to Private tuition.

10) Scheduled Lesson Table

- A) Must display scheduled lessons information by student, in an order and easy to understand fashion.
- B) Must display associated details regarding: Student, Teacher and instrument.
- C) Must be able to easily navigate between scheduled lessons records.
- D) Must be able to search for Student.
- E) Must provide link to Add field form.
- F) Must provide ability to append details.
- G) Must provide a link to Login or Confirmation form depending if no user is logged in.
- H) Must provide a link back to Private tuition.

11) Purchased Lesson Table

- A) Must display Purchased lessons information by student, in an order and easy to understand fashion.
- B) Must display associated details regarding: Student.
- C) Must be able to easily navigate between Purchased lessons records.
- D) Must be able to search for Student.
- E) Must provide link to Add field form.
- F) Must provide ability to append details.
- G) Must provide a link to Login or Confirmation form depending if no user is logged in.
- H) Must provide a link back to Private tuition.

12) Schedule Calendar

- A) Must provide user with the ability to select a specific date.
- B) Must provide access to calendar times.
- C) Must provide link back to Private tuition.

13) Calendar Time

- A) Must provide user with the ability to select a specific time.
- B) Must provide access to Scheduled Classes.
- C) Must provide access back to schedule calendar dates.

14) Scheduled Classes

- A) Must display classes which have been established for the specific time and date selected.
- B) Must provide a link to scheduled lessons table.
- C) Must provide link to Private Tuition.
- D) Must provide link back to calendar times.

15) Login

- A) Must provide admins with the ability to login.
- B) Must prevent unauthorised

16) Add Field

- A) Must provide the user with the ability to easily enter new details (If logged in).
- B) Must validate entered data.
- C) Must Append Data to table if valid.
- D) Must Provide Extensive user feedback, to simplify system.
- E) Must contain a link to previously used form.

17) Confirmation

- A) Must display a confirmation option (If user logged in).
- B) Must provide a link to previously used form.

18) Upcoming lessons

- A) User must be able to view accurate and up-to-date information regarding the upcoming scheduled lessons.
- B) User Must be able to filter information in relations to the selected entity of student or teacher.
- C) User must be able to easily navigate through the displayed records.
- D) User must be able to navigate to Main Menu from this form.

19) General

- A) System must be professional.
- B) System must be user friendly.
- C) System must be operational.

Project Constraints

When attempting to actively implement a new system or operation within a business, one of the most important aspects to both acknowledge and accurately identify are the associated constraints. These constraints are simply various aspects and areas within a business which may cause conflicts with the implementation process, or vice versa. Identifying these possible issues can prove extremely important and beneficial; effectively providing management with the ability to make necessary modifications to either the business, or the implantation process, which ensures the smooth operation and performance of both objects.

To elaborate further, a few of the categories which should be reviewed are: Cost, Staff, Timescale and legislations. This is due to the fact that each individual category will contain important information regarding the reactions between them and the object of the implementation process. Once again, as mentioned before this is extremely important as it provides the management with an accurate representation of the effect which completing the implementation process will have within the business, be this positive or negative. Either way in the event that an unexpected and undesirable outcome is evident, this also provides the management with information to make various modifications, solving the various issues presented.

Costs:

One of the primary constraints which every business needs to take into consideration with new implementations is cost. Overall, this is due to the fact that each business basically operates solely around the premise of money, in the sense that it is necessary for the initialisation of a business, and their continued operations, thus the overall goal is to establish a form of steady income, and accurately monitor and assess various costs, as to prevent failure. Additionally, the financial department is not solely responsible for this constraint. Depending on the object which the implementation process is centred around, related departments will also need to conduct various forms of research and planning as to minimise any negative repercussions from a failed implementation process.

An example of this could be within an organisation which revolves around the development and marketing of mobile devices. The financial department is responsible for effectively monitoring the financial data relating to the business itself and the implementation process, as to evaluate whether or not the business is in a state where it is stable enough to attempt the process with room for error. Additionally, the marketing and sales department may also be responsible for communicating with the customer base, and evaluating whether or not the implementation of a new device would be well received.

In relation to the current business 'The Mitchel School of Music', it would prove beneficial to assess the various financial requirements needed for the implementation of this database system. Excluding the general cost which would be associated with outsourcing the development of this program, the next important aspect to acknowledge is physical hardware and the necessary software needed to protect both the computer system and the program itself. This includes, but is not limited to:

Hardware

- Multiple computer systems, either laptops or desktops to position within each classroom.
- At least 1 Desktop, which would be used as the primary computer system.
- At least 2 printers.
- 1 or 2 external hard drive(s).
- A Wireless router.
- Computer networking hardware.

Software

- Anti-Virus Protection.

- **Networking.**
- **Microsoft office home and business.**
- **Back-up software.**
- **A secured login system.**

Staff

There are a few important aspects which need to consider in regards to the staff present within the school, and any constraints which they may cause with the system. The first point which will be addressed relates to the management of both the system and the program. It is important to understand that the database program can be modified accordingly as to make it accessible from numerous locations; however, it commonly needs to be situated within a central location. To solve this issue, the business in question will commonly situate the program within a central server system which can be accessed from numerous computer systems. This makes for easy transfer of information, and presents easy access to the system. This has been mentioned, because computer systems are occasionally unreliable and need proper maintained regularly.

This simply translates to the fact that the business will need, or at least benefit from having an IT specialist present within the business. Hence it is also important to address the fact that there are numerous methods which could be used to locate an employee who meets this criteria. The first and likely most simplistic method revolves around simple advertising, with the use of the internet and posters it is possible to attract the attention of new employees. The problem with this however is that it can be time consuming and may not have a high success rate. Alternatively, in the event that the business is unable to locate an individual who possess the skills required for the job, they may make the decision to simply train an existing employee. The benefit of using this method is that the business already has access to all of the necessary resources, with the draw-back being that the process takes time and can cost a fair sum of money.

Additionally, another point to address relates to various complications which the implementation of a database system could cause. It is important to address, that while databases are not extremely complex, introducing the system to individuals who have never seen or accessed a similar program before may be at a loss. This issue would commonly be fixed over time, with simple training or the innovation of user manuals and user friendly interfaces. While not the most devastating problem to be introduced, it is still

important to address, due to the fact that it could be the cause for a slow-down of productivity within the business for a period of time.

Timescale

It is important to understand that the implementation of computer systems, and the database information management system will take time. This simply means that the opportune time to complete this process would be during a period in which the business is inactive, or otherwise least productive. For this school specifically, it may be beneficial to complete this process during the summer months, due to the fact that the majority of students will take the summer break, thus leaving the school with a lot of time, and less to handle overall. If not managed correctly the business and the implementation process could both suffer substantially. For example, it may be difficult to work around the implantation process, thus causing difficulties with locating records, completing classes and generally causing a negative effect on the school's reputation and possible income. Alternatively, micromanaging both the school and the implementation at the same time, could greatly extend the period of time which could be taken to complete the process, thus causing more complications.

Legalisation

The final constraint which will be addressed within this section of the document is legislation. As may be implied by the title, this category is used to relate a number of laws and acts which may affect various aspects and operations within the business. These laws have each been organised and established for their own individual reasons, most commonly to ensure that the customers and employees of a business are both safe and satisfied. Overall, this is an extremely important category to acknowledge and reference when reviewing the general workings of a business, due to the fact that these laws and acts could have serious repercussions on the business, in the event that one is breached. This could cause serious issues for the business's reputation, finances, and may even be the indirect cause for the failure of a business. The laws which will be elaborated upon within this section of the document are as follows:

The Data Protection Act 1998

Data and information are two resources that are extremely important within any business, the information that they collect can be personal or financial data. If used correctly, this data can help improve businesses and help them operate more efficiently, yet in the wrong

hands serious harm can be done to the data subject. This is why the data protection act was brought into commission.

The purpose of this act is ensure that all personal information that is collected by an organisation such as Mitchell's School of Music, is kept safe and used to appropriate use. This act can be split into 8 separate principles overall, they are:

- "Personal data must be obtained and processed fairly and lawfully"
- "Personal data must be held for specified and lawful purposes"
- "Personal data must be adequate, relevant and not excessive"
- "Personal data must be accurate and up-to-date"
- "Personal data must not be kept for longer than necessary"
- "Personal data must be processed in accordance with the data subject's rights"
- "Personal data must be kept securely"
- "Personal data must not be transferred from country to country without protection"

The principles above are used to ensure that data and the data subject are protected. The data subject has the right to ask for compensation if the business discloses information unauthorised, or if the information held is inaccurate etc. They can also demand to have their information back whenever they want.

Mitchells School of Music would have to be very careful with this law, breaking it could harm the business drastically. To avoid breaking it by accident, Mitchells School of Music needs to ensure that they keep the personal data that they have collected safe and protected. Even if Mitchells School of Music had no involvement with a leak of information, they have full responsibility. This is an issue because as can be found within the case study, it mentioned that currently, staff will leave important information on sticky notes. If Mitchells School of Music was to store their information on a database, they would have to ensure that it would be secure through the use of login protection. This is needed to ensure that unauthorised users cannot access the documents.

Copyright, Designs and Patents Act 1988

There has always been a problem with the prospect that people could steal someone else's original work and put their name on it. This is why the copyright designs and patents act was introduced. Overall it was created with the intent to protect original material from being copied, ensuring that the creator gets all credit. It was passed in 1988 with the goal to protect the investment of money, time and effort that the creator had to put into the item. A few relatable examples of instances when this is in use are with games, music and software etc.

The two main objectives of this Act are to ensure that people are rewarded for their hard work and give them protection if someone tries to steal their work.

Mitchells School of Music would have to be cautious with this act, particularly when creating advertisement material. If they use images or music that is copyrighted by another company or person, they will be breaching this law. This would be unsatisfactory primarily because they would likely be charged a large fee.

Computer Misuse Act 1990

Typically today, most people own a computer or at least known how to use one, they are beneficial because they provide support in completing several tasks. However, there are several instances where users will use computers to harm, steal or disrupt others. An example of this is when users create harmful software that is either designed to infiltrate a user's computer and steal personal information, or one that's only purpose is to cause mayhem or even destroy documents within a computer. Before 1990, these acts were completely legal; which is why this act was finally brought into commission, as the government finally decided to clamp down on hackers.

There are three new offences that this Act is aware of:

- The unauthorised access of computer material - Stealing someone's password and username.
- Unauthorised access with the intent to commit or facilitate a crime – Stealing someone's password to access bank accounts etc.
- Unauthorised modification of computer material – A good example of this is viruses; these pieces of software will infiltrate a computer and can then do several things such as damage or delete files on the computer.

It is important for Mitchells School of Music to ensure that all staff are trained in the fact that they should not open random suspicious emails or downloads, as they could bring a virus into the business, which could compromise documents and systems.

Health and Safety Act

The health and safety act was introduced with the intent to ensure that both employees and customers of a business receive the care and protection that they deserve. This act affects the operations of a business by ensuring that businesses provide safe products, from a safe and clean environment.

For Mitchell's School of Music, the main concern that has come to my attention is farming equipment. Farming equipment can be extremely dangerous to work with, and can be the cause of several accidents. Mitchells School of Music would have to find ways to ensure that employees minimize risk, such as making sure that all staff are qualified and know standard safety rules. The same can be said for the vets within the other side of the business, another important point for them being that they must ensure that their working environment is clean and safe.

If Mitchells School of Music does not follow the set of rules set by this act, they will break the law and may be the external cause for someone getting hurt, which could lead to an extreme court case costing the business reputation and money.

Project plan

Task Number	Task	Start Date	Comment	Completed Date
Planning				
1	Review current information management system.	September 28 th		October 12 th
2	Conduct an interview with the owner/manager.	October 12 th		October 14 th
3	Design and distribute questionnaires.	October 16 th		October 25 th
4	Review collected information, then establish user system requirements.	November 02 nd		November 06 th
5	Identify necessary and desired tables, queries and designs.	November 07 th		November 12 th
Design				
6	Design Data Dictionaries	November 12 th		November 15 th
7	Design General Table structure	November 15 th		November 17 nd
8	Design Queries	November 18 th		November 20 th
9	Design Splash Screen	November 21 st		November 22 nd
10	Design School Menu	November 29 th		December 01 st
11	Design private-school tuition menu	December 1 st		December 3 rd
12	Design standard table forms (Students, Teachers, Instruments, Grades, Lesson bundles)	December 4 th		December 8 th

13	Design Additional table forms (Scheduled Lessons, Purchased Lessons)	December 9 th		December 13 th
14	Design schedule timetable	December 14 th		December 16 th
Implementation				
15	Create Data Dictionaries	December 16 th		December 20 th
16	Create General Table structure	December 20 th		December 22 nd
17	Create Queries	December 22 nd		December 30 th
18	Create Splash Screen	January 1 st		January 3 rd
19	Create School Menu	January 12 th		January 14 th
20	Create private-school tuition menu	January 17 th		January 18 th
21	Create standard table forms (Students, Teachers, Instruments, Grades, Lesson bundles)	January 20 th		February 13 th
22	Create Additional table forms (Scheduled Lessons, Purchased Lessons)	February 14 th		February 21 st
23	Create schedule timetable	February 21 st		February 26 th
Testing				
24	Test Tables	March 01 st		March 10 th
25	Test Table Display Forms	March 11 th		March 16 th
26	Test General Forms	March 16 th		March 23 rd

27	Complete Test Plans	March 23 rd		April 01 th
28	Complete Test Evidence	April 01 th		April 22 nd
Evaluation				
29	Complete Evaluation			

Methodologies

A software programming methodology could simply be identified as a structure plan for development of a project. This is done by initially reviewing the various steps which must be completed to effectively produce the end product. To achieve this goal, these steps would be categorised within various 'phases' each of which can be provided with individualised goals, objectives and tasks, depending on the methodology in question. It would also be common for the completion order of these phases and their tasks to change in correspondence to the selected methodology.

The use of methodologies within the development of software programs can be deemed extremely important and beneficial. This is commonly due to the fact, that it in segmenting these processes, they are effectively simplifying the management of physical and human resources, while also aiding within the organisational process of reviewing and establishing tasks.

A few of the methodologies which will be reviewed for the project are:

Dynamic systems development method (DSDM)

Definition

Dynamic Systems Development Method, also regarded as DSDM could be identified a agile project development framework, typically used within the construction and publishing of software products. It is interesting to note that this development methodology was initially conceived under the intention of establishing discipline to the Rapid Application Development (RAD) method, yet quickly became a generic choice of framework for the management of new projects.

In regards to what this Methodologies purpose and general aims are, it has been suggested that DSDM is an iterative and incremental method, which actively embraces the principles of Agile development, including those regarding continuous user/customer involvement.

DSDM would begin by establishing fixed cost, quality, and time, and would then use the

MoSCoW prioritisation of scope, to separate the project into the following categories: "Musts", "Should", "Could", "Wont", and "Have". This is done to actively ensure that the stated time constraints can be managed and met accordingly.

Core Techniques

Time Boxing – This technique would be used within DSDM to actively support the primary goals, and to identify the current state of the project – as to ensure that it is one time, budget, and is currently expressing the desired quality. To achieve this, the primary function of time boxing is to segment the project into various portions, each of which will be fixed with a specific budget, and delivery date. For each of these portions, the MoSCoW technique would be used to identify and assign a number of requirements. This is an interesting practice for the fact that the primary variables present within each portion are time, money and requirements, with the latter being the only variable that is not fixed. This essentially means, that should if a project is running low on money or time, the requirement with the lowest priority will be void. It is important to address however, that this does not mean that the development process would be completed with a unfinished product, due to the PARETO principle, which states that 80% of the project comes from 20% of the systems requirements. This essentially means that as long as the extensively important 20% of the requirements are met, then the system will meet the business's needs.

MoSCoW – This term is used to simply represent a prioritisation system of relevant items, for which would be requirements in the context of DSDM. It is an acronym for the following:

- MUST establish this requirement as to meet the business's needs,
- SHOULD have this requirement, however it is not essential for the success of this project,
- COULD have this requirement in the event that it does not negatively affect the project,
- WON'T would be used to represent a requirement that has been void by the stakeholder consensus that it will not be present within a given release, however it may still be taken into consideration at future dates

Work Shop – The work shop could simply be identified as an event in which different stakeholders for the project will be gathered together, as to discuss the requirements for the system or business area currently in development.

Rapid Application Development (RAD)

Rapid Application Development, also known as RAD could simply be identified as methodology used within the development of software, which was developed with the intention of placing a heavier focus on rapid prototyping, over that of extensive planning. On this note it would likely be beneficial to address the fact that a prototype is a working model of the project, which expresses an representation of the end products purpose, goals and contents.

When using this model, functional models would be developed in parallel to the various prototypes mentioned, and would later be integrated to complete the product in less time, which therefore translates to faster product delivery. It is interesting to note, that due to the fact that there is extremely small amounts of planning completed, this makes implementing new changes within the development process much easier.

Teams developing products under this model would commonly consist of small collections of individuals, typically consisting of developers, domain experts, customer repetitive and other IT resources, who would work on the their component or prototyping.

The advantages of the RAD Model are:

- ❖ Changing requirements can be accommodated.
- ❖ Easy to measure progress.
- ❖ Increases reusability of components.
- ❖ Iteration time can be short with use of powerful RAD tools.
- ❖ Actively encourages customer feedback.
- ❖ Productivity with fewer people in a short time.
- ❖ Reduced development times.
- ❖ Quick initial reviews occur.

The disadvantages of the RAD Model are:

- ❖ Dependency on technically strong team members for identifying business requirements.
- ❖ Increased management complexity.
- ❖ Only system that can be modularized can be built using RAD.
- ❖ Requires highly skilled developers/designers.
- ❖ High dependency on modelling skills.
- ❖ Inapplicable to cheaper projects as cost of modelling and automated code generation is very high.
- ❖ Suitable for systems that are component based and scalable.
- ❖ Suitable for project requiring shorter development times.

Waterfall

It has been established that the waterfall model was the first process to be introduced to the planning and development of software programs. In regards to the overall structure and purpose of the model, it has commonly been regarded as being a linear-sequential life cycle model, indicating that it possesses a simplistic structure, one which progresses in a sequential manner. To effectively provide a better understanding of the model and its purpose, it is initially important to review the various phases present within it. This can include the following; however it is commonly open to interpretation and could therefore vary mildly from program.

Possible phase structure

- ❖ *Requirement gathering and Analysis –*

This phase is used to effectively analyse all possible requirements regarding the pre-developed system. Any information collected from this process would then be stored within a requirement specification document and stored accordingly.

- ❖ *Interviews, observations, and Questionnaires –*

This phase is commonly used to acquire information which can be used to effectively develop user requirements in association to the program. This can be done using a method similar to one mentioned in the title, and would be documented accordingly for reference and general guidance when developing the program.

❖ *System Design -*

The information collected within the first stage would be further analysed within this phase, and preparations for the overall designing of the program would begin. This is done to help provide an accurate representation of the system requirements which would be needed to identify the hardware and system requirements which would be needed to operate the program. In addition to this, it has also been known to help establish a system architecture.

❖ *Implementation -*

Using input from the system design stage, the development of the system can begin, however this is achieved by developing smaller programs commonly regarded as units. Each of these units is assigned a specific goal which they must achieve. To ensure that this is met, they will be tested within the next stage before being integrated.

❖ *Integration and Testing –*

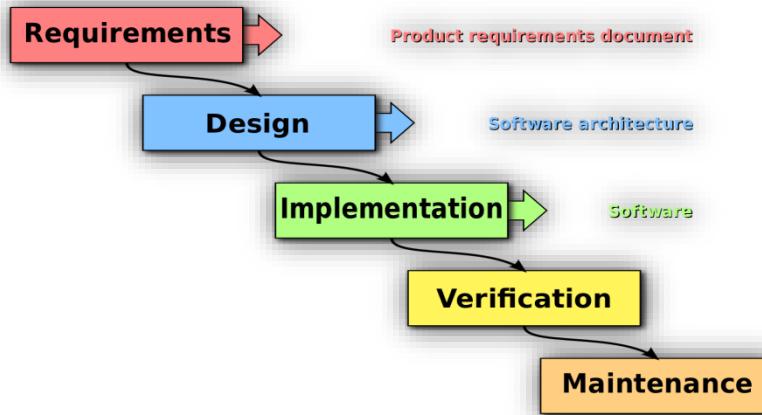
As mentioned, this stage is responsible for the testing of each individual unit, ensuring that they meet their goals effectively. Once this has been established, it is possible to begin integrating them into an overall system. Upon integrating a new unit, it would be common practice and generally beneficial to complete a test of the entire system, as to ensure that no complications where imposed by the integration process, and that the system runs appropriately.

❖ *Deployment of system –*

This stage could simply be regarded as the end goal for the system, for once all implementation and testing has been completed, and a stable and complete system is developed. The ability to release it to a market or customer environment becomes available.

❖ *Maintenance –*

Upon releasing the system within a customer environment, it is likely that various issues and oversights will become apparent. To solve these issues, the development team will need to append these issues with the use of patches. Additionally, it may be deemed beneficial to modify the overall system drastically, as to improve performance and make it more effective at meeting its purpose. This modification would commonly be remarkedeted as an improved variation of the system.



Interestingly, what makes this model unique is the fact that all phases which are established pre-development of a system or program must be followed sequentially, and must be completed before advancing. This simply means that a business using this model, would need to meet all aims and objects set within the system design, before they are able to begin working within and planning the implementation process. In addition to this, once a phase has been completed and thus passed, it commonly cannot be returned to until the overall process is completed. Intriguingly, as the name may imply, there have been comparisons between this model and a waterfall. This can be seen when analysing the progression of a waterfall, once it has begun descending, the water is unable to turn back.

Advantages

A general and primary advantage which is commonly associated with this model relates to organisation and general management of a project. In advance to elaborating further, it could prove beneficial to address the base reasoning for these points. As has been previously mentioned within this document, the general structure for this model revolves around the linear sequential completion of phases. Already this simplifies the management process, and establishes a clear organised structure and working order within the completion of this project. Then further with the individualised evaluation and creation of each phase, provides management figures with the ability to thoroughly consider and establish the necessary aims and objectives which need to be met within the completion of the phase as to meet the set requirements, and thus ensure the system runs at the predicted and preferred performance.

It is also beneficial to note that due to the fact that the structure of the waterfall model restricts all members of the team working on this project to partake within the same phase

until completion, there are numerous associated advantages. One such advantage is that because all members are working around the same goal, not only in theory should that goal be met soon, it should also be met with better efficiency and accuracy. This should also make it much more simplistic to understand and compile ideas and work relating to the phase, while also making it easier to manage the team.

A few of the advantages which are commonly associated with the waterfall model are as follows:

❖ *Simple and easy to understand –*

This model has proved extremely simplistic and beneficial to use, likely due to its simplistic and well organised structure. This lack of complexity makes it possible to review the model and easily understand the various phases, and the key aims and objects involved.

❖ *Easy management -*

It has commonly been expressed, that the simplistic and strong structure of this model, helps simplify management of the system and overall project. This is due to the fact that all members would be organised within teams to complete work within the same area. This results in less diversity and thus less complications generally.

❖ *Each phase has specific deliverables and a review process -*

Every phase which is established within the planning process is done with a set of aims and objectives. This is beneficial, as it helps establish an effectively clear plan regarding the general purpose for the phase. This thus provides a form of guideline for the members of the team, which could be used for reference or guidance. In addition to this, each phase is completed with a review process. This is beneficial as it helps evaluate that the phase was complete effectively and correctly, before advancing to the next phase.

❖ *Phases are completed once at a time -*

As previously mentioned, the team constructing the system will systematically progress from one phase to the next, once the designated aims and objectives have been met. This is beneficial in the sense that it creates a form of unity, as all members will be working together on the same phase, which in theory should lessen the time needed to complete phases, and improve general productivity. In addition

to this, as previously addressed, this simplifies operations, and this makes it easier for management to complete their assigned jobs.

❖ *Beneficial for smaller projects -*

This model has been deemed more beneficial for smaller projects, as it means that the team constructing the system will need less time evaluating and complete each phase accordingly. This also means that their resources can be used to more accurately when completing these phases, this in theory improve the quality of completion. In addition to this, it also helps ensure that all requirements are understood clearly.

❖ *Clearly defined stages -*

When planning and implementing the various stages needed for the completion of the desired system, there is commonly a lot of thought and consideration. This simply enforces the fact that these phases, along with their aims and objects are well structured overall. This is beneficial as

❖ *Well understood milestones -*

As previously mentioned, the various stages used for this model are thoroughly defined, and thus have been established in a manner which makes them easy to understand and follow accordingly. This can prove extremely beneficial, as it reduces the chance of misunderstandings between members of staff working on the project, effectively improving productivity.

❖ *Easy to arrange tasks -*

Due to the overall simplified management process, and general unity within the workplace, management may commonly find a much easier time assigning and organising tasks.

❖ *Processes and results are well managed -*

This model ensures that time is taken at the end of each phase for the evaluation of the overall phase, and the documentation of these results. This is important, for there are numerous pieces of information which can be effectively used within other phases to complete various functions, and for reference later within the production process.

Disadvantages

As has been previously mentioned, this model's structure revolves around the sequential completion of pre-established phases. While this can prove to be one of the largest benefits for this methodology, it is also the cause for a few of its largest limitations and general disadvantages. For example, due to the fact that each phase must be complete thoroughly and individually, the completion of a project can take extremely long periods of time, typically in correlation to its overall size. This becomes problematic for numerous reasons, one of which being that the team will not be able to establish a functional system until late within the development process.

In addition to this, it should be known that one of the most important processes conducted within the planning stages of system development relates to the collection and management of requirements for the system in question. This is due to the fact that the system will typically be built around these requirements, being modified accordingly as to accommodate them. This is important to note, for this model has been known to poorly accommodate requirements in correlation to the time period between the beginning and end of the development and implementation of the system. There are various instances in which requirements for a system can change significantly, during these later processes, however due to the overall structure of the model, once the requirements have been locked in, the system will need to be built around them no matter what, with the alternative being to scrap the project totally.

One final note to make in this regard revolves around the constant technological advancements over time. This can also prove quite damaging, due to the fact that these projects would be initially planned and organised with the current hardware and software for that time. Yet in the event that the completion of this project takes even a year or more, there is a high likelihood that it would already be outdated, and operating on outdated hardware.

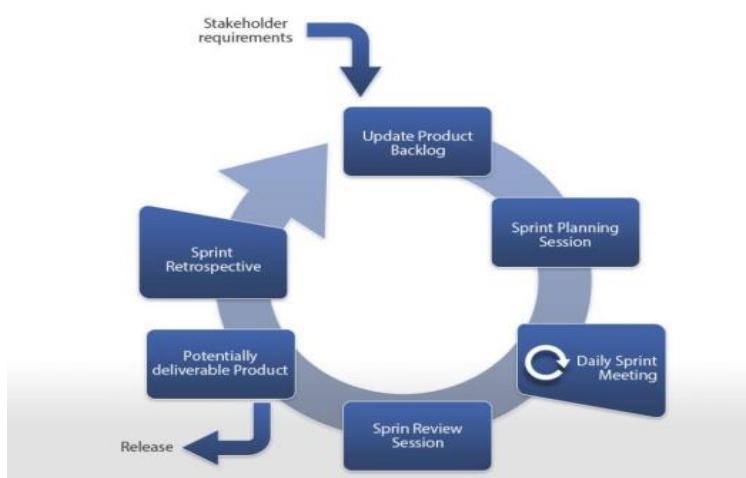
To conclude this segment, it would be beneficial to address the general disadvantages which are associated with the overall structure of the waterfall model. As has been touched upon, one of the main issues relates to the fact that users cannot revisit completed phases, which can prove a disadvantage for numerous reasons. In addition to this, all members of the team developing the project are restricted from continuing at their own pace, and may thus be made to work within areas which they do not excel in, or find uncomfortable.

A few specific disadvantages are as follows:

- ❖ *Quite ineffective for complex and object-orientated projects –*
This model can prove very ineffective for projects which fit the criteria listed in the title, for these projects may require consistent updating between stages, or are merely too complex and this time consuming to complete using Waterfall.
- ❖ *For long and on-going projects this is a poor model to use –*
As previously mentioned, this model has been deemed ineffective for large and complex projects, due to the amount of time which could be needed to complete them completely. In addition to this, attempting to implement this model into an on-going project could prove ineffective due to the fact that many phases would already be passed, and would cause various complications in regards to the order.
- ❖ *Not suitable for projects with a high-likelihood for change, thus creating risk and uncertainty –*
As has been mentioned, a lot of the requirements for the project will change over time; be this standard operational requirements, or hardware requirements. These changes can cause extremely negative side-effects for the outcome of the project is completed. In addition to this, changes to the overall structural plan of the project could prove to be a disadvantage; due to the fact the completed phases cannot be re-evaluated. Thus the waterfall model should not be used for projects with a high probability for change.
- ❖ *Difficult to measure overall progress within stages –*
There have been various instances in which users have expressed difficulties in measuring the total progress made in relations to each stage, and this the overall project.
- ❖ *Is unable to accommodate a change in requirements –*
Projects using this model are unable to accommodate changes made to the requirements.
- ❖ *Integration is completed near the end of this process, thus meaning that technological, business bottle-necks or challenges cannot be identified early –*
An issue present within the established order for the stages is that integration is completed very late within the development process's life cycle. This could prove to have extreme negative consequences of the project, due to the fact that various technological and business related issue would become apparent within this process, thus leaving management with little time to react accordingly.

SCRUM

SCRUM is an agile project methodology which was initially established as an alternative to the waterfall model, with a larger focus around the management and development of larger and more complex software projects. There are many different features present within this methodology which help to individualise and contrast against the more tradition model – waterfall. As has been previously mentioned, one of the main issues present within the waterfall model relates to the fact that it was unable to manage and compensate for changing requirements imposed by the users. To counteract this issue, SCRUM was designed around these changes, with an example of such being that user feedback is actively encouraged between each stage of development. It has been designed in prediction of these changes, making it more flexible and thus able to handle them accordingly, effectively modifying the program to better represent the customer's needs.



Scrum Teams

The next important section to address relates to the usage of this methodology and how teams should be structured. These SCRUM teams would commonly consist of a number of distinctive roles, including:

- ❖ *Product Owner*

The product owner of the team would simply be identified as the senior team leader, and is responsible for delivering the clients wishes in regards to the software projects.

❖ *Scrum Development Team*

As the name of this general role implies, it would commonly be dawned by a team of 6-8 multi-disciplinary individuals. This includes developers, testers, and business analysts who are responsible for the development and testing of the project.

❖ *Scrum Master*

The scrum master is assigned the responsibility of managing the Scrum teams during the daily “Scrum meetings”. In addition to this, they are also responsible for resolving any issues which become apparent, setting coding standards, and enforcing various deadlines.

Product Backlog

When using the Scrum methodology, the software project in question will be systematically broken down into functional parts, each of which are then broken down further to function tasks, by the Scrum team. It is important to note that these functional tasks are commonly identified and generally referenced as Product Backlog items, and would then be organised in relations to priority, this organised list being known as a product backlog list.

These backlog items would commonly consist of:

❖ *Title*

A short tile which relates to the statement, used to simply identify the item.

❖ *Statement*

A concise statement used to detail the overall functional requirements of the system. This would be portrayed from the view of a user as a user story.

❖ *Criteria*

This is simply a short line of text used to identify the overall criteria needed to make this system requirement operate.

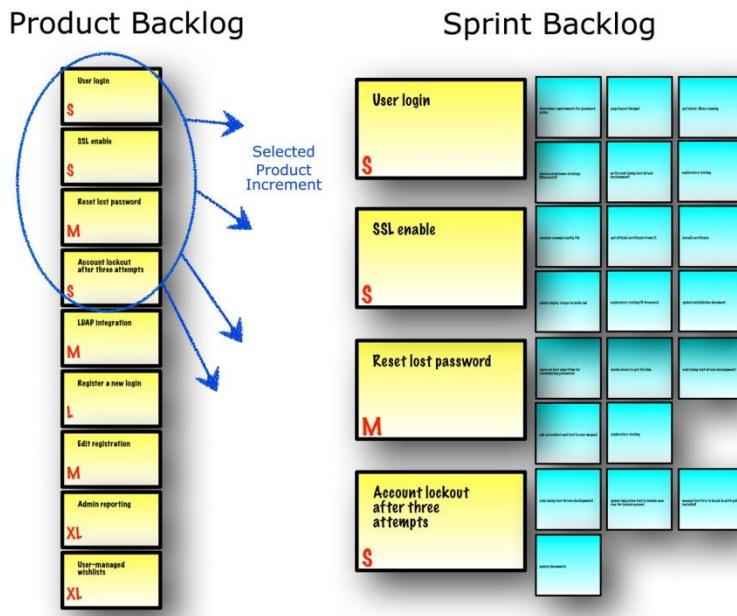
❖ *Resource Estimation*

A simple estimation of the resources needed to complete this PBI, including physical resources and employees.

Sprint Backlog

Occasionally, a day is designated as a Sprint Planning meeting, where the scrum team must review the various product backlog tasks, before deciding which tasks must be worked on during the next sprint development cycle. During this process, the PBI's are

transferred from the product backlog to the Sprint backlog. An example of this can be seen below:



Sprint backlogs are broken down into tasks, and then subtasks accordingly.

The Sprint

Once the planning and assigning of tasks has been completed, a sprint will typically take place. This is simply a time boxed event used to effectively complete one or more user stories. It is extremely important to note however, that user stories must be completed within the sprint time box, otherwise it could cause various technical difficulties, such as a delay on the overall project. Should this be the case, it would be the scrum master's responsibility to allocate more people/ time and resources to the task in question.

During the sprint period, the scrum master would commonly be responsible for the process of organising daily team meetings known as daily scrum. These are an interesting form of management and organisation, for they will commonly span across a 15-minute time period, in which each member of the team will stand up and address the work which they have completed within the last 24-hour period, and identify the work which will be completed within the next 24 hours. As briefly touched upon, these can prove extremely beneficial, due to the fact that they provide the ability to effectively and accurately monitor and analyse the progress made with specific tasks, identify problems which are responsible for blocking continued progress of the task, while also assessing the various steps which could be taken to solve the issues. Additionally, they are also beneficial for helping

focus the scrum team, directing them with specific goals, while also encouraging them to complete all work within the designated time frame.

Once the sprint has been concluded there would commonly be a sprint review meeting held. This is effectively used to formally review the total progress which was made during the sprint, and would be attended by the product owner, scrum team, and end users. In regards to the overall purpose of this meeting, it helps provide a demonstration of the operation software which is has been thoroughly tested and stable, thus making it ready for release. Additionally, having the end users present is extremely important for it provides the team with the ability to acquire updated feedback, which would then be used to make future modifications accordingly. During these meetings, the product owner is required to identify the current items which are totally completed, the items which could use amendment from the user feedback, address any newly recognised needs for the user, and the items which are not ready for release, and must therefore be returned to the product backlog.

Finally, another important meeting which would commonly be established for teams using Scrum is the sprint retrospective meeting. This would be attended by the scrum master and the team, and would commonly be used to review and reflect upon overall performance and progression within the previous sprint. In other words, this meeting could simply be identified as a self-evaluation for all team members, as to identify any weaknesses or issues, and to improve, or remove them for future sprints. This is important, as it can help improve efficiency and quality of work completed during these sprint periods.

Monitoring Progress

The scrum master is assigned the responsibility of monitoring and managing the scrum backlog, and the assigning of tasks to members of the team. To effectively complete this job, they would commonly use a backlog chart, which will be accurately display the progress for each assigned task. It addition to this, it would commonly contain information regarding the resources and “time box” which has been designated for each individualised task. The elaborate further, the time box is simply the time limit which has been establish for the task at hand.

One final note to make on this regard, is that during sprint backlog meetings, this information would commonly be analysed and used to decide whether the assigned resources and time limit for a specific task need to be modified in due to the current

progression. This can include adding additional personnel, extending time limits and simply adding resources.

Charts

Specific charts, commonly identified as burndown charts are commonly used within the monitoring progress of the project. As may be implied, these are simply visual charts which are used to accurately display the number of hours which are left to the dead-line for each individualised sprint task. This chart will also need to be regularly updated, as to ensure that it accurately displays the tasks which are currently active, and not completed.

Interestingly, it is important to address that while the overall duration of the sprint may be time boxed at approximately 30 days, the time being used can change in accordance to progress within the sprint tasks, meaning that it can be increased or decreased appropriately. Overall, using this feature proves extremely beneficial, as it provides the scrum master with the ability to accurately track, add and modify tasks accordingly. In addition to this, they can also easily transfer employees from tasks to task, or assign additional resources in general.

Advantages

The scrum methodology is commonly regarded as being extremely advantageous to use within various software projects, especially for those that are more complex. It is important to acknowledge that there are numerous reasons for this, with the common consensus being that in comparison to other models such as waterfall, it is much more capable of handling changes to the project, and encourages feedback from their end users. It has been noted that scrum is effective for developing programming projects within a scheduled time period, and that it may be more cost efficient to use than that of other models.

Due to the overall structure for the scrum methodology, teams are provided with the ability to effectively and swiftly create complex developments in short periods of time, due to the accelerated coding and testing processes. In addition to this, as it is a lightly controlled method, the projects developed within are exposed to frequent updates and changes making it much more possible and simplistic to keep up with new user requirements, and easily rectify any errors previously made. In regards to the overall management of the project, as previously mentioned, the scrum method encourages the

establishment and holding of regular meetings on various occasions. These meetings provide the ability to easily review all important figures and progression made in relations to the project, and provide the ability to easily establish and assign tasks, aims and objectives, providing clear organised operations. In addition to this, the daily meetings which are held are extremely beneficial as they provide the ability to easily identify various issues far in advance, thus providing adequate time to address and prevent them.

To break this down however, the advantages are as follows:

- ❖ *Cost efficient compared to other models.*
- ❖ *Better identification of issues.*
- ❖ *Easier to meet project deadline.*
- ❖ *Short sprints make updating the project easier.*
- ❖ *Lightly controlled method encourages organising through meetings.*
- ❖ *Development is quick, professional and more simplistic.*

Disadvantages

One of the main disadvantages associated with the agile Scrum, is that it has been identified as one of the leading causes for “Scope of creep”. To elaborate further, this “Scope of creep” is used to identify project management which is experiencing uncontrolled changes, or continuous growths within the project’s scope. This typically occurs in the event that the overall scope of the project is not properly defined, documented or controlled, and is generally considered very harmful. Should an end date not be clearly defined, the project management stakeholders may feel tempted to continuously update requirements and make demands in regards to new functionality.

An additional note is that regards to structure and organisation, it has previously been mentioned that Scrum relies quite heavily on the management of physical and human resources in correlation to assigned tasks. In the event that these tasks and other figures are not properly monitored, managed and documented, the overall accuracy and progression within tasks will decline, thus causing planning and production to span across numerous sprints.

In addition to reviewing the issues in relation to organisational structure, it is also extremely important to analyse the team make-up. It should be acknowledged that the Scrum

methodology would typically require a small team, one which should consist of experienced and cooperative members. Due to the team size, it is commonly recommended that this methodology would merely be used to complete smaller projects. There are various issues which connect to these various points, for example, establishing a team of individuals who do not possess the necessary experience could cause issues during the sprint, where some members may not be pulling their weight, while if the members do not cooperate, operations and efficiency could fall through.

To break these points down further:

- ❖ Cause of Scope Creep.
- ❖ Undefined tasks equal poor management and organisation.
- ❖ Larger projects can conflict with team size.
- ❖ Lack of experience within team may lead to decline of efficiency.
- ❖ Poor cooperation between team may lead to decline of organisation.

Extreme Programming (XP)

Extreme programming is an agile methodology which was established as an alternative to the waterfall model. This method was established under the concept of providing developers with the ability to develop software projects with a heavy focus on the improvement of software quality, and overall responsiveness to changing customer requirements. In regards to operations, in an attempt to reduce total costs associated with changes to requirements, development would be broken into multiple shortened cycles, in alternative to an individual extended cycle. As previously touched upon, operating under these ideals results in modifications around the projects designs, requirements and other areas become natural, and maybe even desirable. This simply results in a situation where modifications will be planned for, instead of attempting to establish a stable set of requirements.

Interestingly, the author of the extreme programming methodology has made statements and general comments about the method, including the fact that the most important aspect is the code, and should therefore be prioritised. However, he also emphasised on the belief that a written test must be produced in advance of creating code, as to know whether or not the code succeeds at completing a specific task. Finally, he expressed the idea that the code within projects would benefit from cooperative development in pairs, as

this can stimulate new ideas and concepts when comparing and expressing the developed code to each other.

One final point to address within this area briefly touches upon the operations and structure for this methodology. The tradition structure and development process for a software project commonly revolves around the assumption that software must hit the desired mark the first time, as this would be the most economical approach overall. Extreme Programming attempts to oppose this standard, and revolves around the early initialisation of a project, effectively building something real which operates within a limit way, and can be fitted into an overall structure which will be built at the convenience for the further code to be implemented, opposed to the alternative ultimately completed system. Another important fact which should be taken into consideration is team structure. Unlike other methodologies which typically have a well-planned team structure in advance which, effectively segments tasks and jobs in association to specific operational areas. Extreme Programming however, basically disregards this overall structure, and instead encourages all members of the team to complete tasks in relations to the development, testing, analyses, designing and continued integration of code and project in general. As a final closing note for this specific topic, it is beneficial and generally important to acknowledge the fact that this structure effectively establishes a more face-to-face communication process, and therefore will commonly remove the need for extensive and advanced storage and communication of informational documents.

Activities

As previously mentioned within this section of the document, Extreme Programming consists of four basic activates which effectively make up its operational structure, including; coding, testing, listening, and designing.

❖ *Coding*

As has been previously mentioned, advocates for Extreme Programming commonly emphasise on the belief that the production of code is one of the most important products created through a projects development. This is generally due to the fact that without the code, there will be no operational product to test, compare or review. This code is highly regarded for it can typically be used to find a suitable solution in most scenarios. Additionally however, there have also been instances in which the software code could be used to effective communicate complex ideas and general thoughts associated with complex problems. Accurately expressing ideas and concepts can be

extremely difficult within software development especially for some extremely complicated operations; therefore some developers may find it much more simplistic and beneficial to express their ideas through the use of code, as to effectively demonstrate their motives and goals. Ideas and thoughts can be interrupted in numerous ways, while code is commonly portrayed as clear and precise, effectively implying that code will be interpreted in a singular manner.

❖ *Testing*

Extreme programming addresses testing in an interesting manner. It has been expressed that testing would be viewed in the sense that if a small completion of testing can identify and eliminate a few flaws, this must mean that a large quantity of tests will be able to identify and eliminate a much large quantity of flaws. In actual practice there are a few different methods which can be used depending on the specific set of circumstances. This includes “uni-tests”, “Acceptance tests”, and “Integration testing”. To elaborate further, “Uni- tests” are the individual and extensive testing of the code of a specific feature or operation within the project. In completing this activity, the programmers are tasked with the planning and development of numerous automated tests which actively attempt to push the piece of coding hard in different areas, using various methods. This is done to effectively tests whether or not the code will break under a number of stressful or unique tests. This is beneficial in the sense that should the code break, the developer(s) are provided with information regarding why it broke, and maybe how they can change it. Alternatively, should all testing return positive feedback this implies that the coding is completed. Additionally, it should be acknowledged that all coding revolving around the project must be thoroughly tested in before advancing between features.

The next testing method to address is “Acceptance tests”, which are simply conducted as a means to ensure that the current user and therefore system requirements are accurately understood by the developers in a state which will satisfy the customer’s actual requirements. This effectively ensures that the developed coding has been created to accurately meet customer requirements.

The last method which will be reviewed within this section is “integration testing”, which was initially conducted system-wide as a daily activity. This was beneficial for it provided the ability to maintain an updated review and status of the project, and also

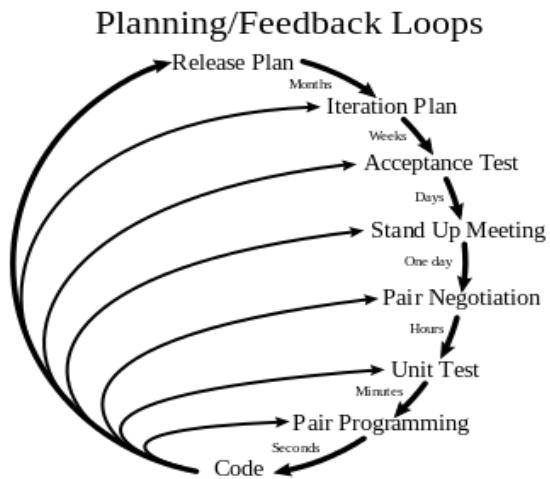
essential within the detection and identification of incompatible interfaces which could be reconnected accordingly. In the event that these tests were neglected, it is possible that the separate sections would eventually diverge widely from coherent functionality. Over time however, a decision was made to reduce the overall frequency of this testing process, limiting it to a weekly activity or less depending on the current stability of interfaces within the system.

❖ *Listening*

Listening is identified as an extremely important process within the planning of software projects. This is generally due to the fact that the programmers must listen to the customer requirements and needs for the system, effectively detailing what the system must accomplish, and the “business logic” which would be needed. These requirements must be understood clearly, as to provide the customer(s) with accurate feedback about any technical aspects regarding whether or not various problems can be solved and how.

❖ *Designing*

While the extreme programming methodology attempts to operate with little to no structural and operational planning, focusing instead on the alternative aspects; Coding, Listening and designing, it will be limited in terms of how far it would go. This is because the system will become more complex as it is developed, a limitation which will correspond to the projects overall size. This can have extremely negative effects on the quality of the project, and the efficiency of developers as dependences within the system cease to be clear. To rectify and avoid this limitation completely, it would be recommended to develop an operational structure which effectively organises the systems logic. Establishing a proper design will help avoid dependencies throughout the system, resulting in a scenario where making modifications to a specific parts of the system without affecting others.



Values

❖ *Communication*

As has been expressed and mentioned previously within the document, perhaps one of the most important aspects and processes within the development of software projects would be communication. This is especially evident within the planning and establishment of user requirements for the project in question. In standard software methodologies this would typically be accomplished with the use of documentation; however extreme programming focuses on the rapid project building and transfer of information between members of the development team. The overall goal for the communication process is to then compile requirements for the system, and effectively ensure that all members of the team are provided with a shared and consistent view of the system, ensuring that these requirements are match the user's view of the system. This all revolves around the general practices and simplistic designs which are favoured within this method, along with the common metaphors, collaboration for users and programmers, feedback and the use of verbal communication.

❖ *Simplicity*

The extreme Programming methodology encourages general simplistic when developing software projects. An example of this would be within the initialisation of the project, where developers are encouraged to start a simplistic interpretation of the project, and then adding additional functionality later within development. Interestingly, when contrasted against more traditional methodologies, extreme programming insists that programming focus should be on the current needs and requirements rather than those of the future.

It is beneficial to address that this is likely due to the fact that Extreme Programming opening encourages and expects the consistent modification of requirements, thus making it near impossible and a liability to attempt to establish and meet long-term requirements. This has been identified as a disadvantage to some degree, for it can increase the total amount of work which must be completed each day, modifying the system accordingly. It has been argued however, that this is countered by the advantage of not needing to invest within future requirements which could change, causing them to become irrelevant. Creating code and designs for the development of uncertain future requirements indicates the general risk of distributing resources on aspects of the project which may not be needed.

A final point to address is that this simplicity will affect the overall communication value, as the simplicity in design and coding should therefore improve the quality of the communication. This is due to the fact that a simple collection of code, with a simple design will be much easier to understand.

❖ *Feedback*

Feedback is an important aspect present within Extreme Programming in multiple instances and scenarios. An example of this would be within the creation, testing and eventual integration of a unit test, which effectively provides feedback on operational status and an brief evaluation of how the system handles the integration. Alternatively, feedback can also be collected by the customer(s) of the specific system. This would be done with the use of various functional tests (known as acceptance tests) which are designed by the customer(s) and testers. These tests would be used within the development process to ensure that the system can handle unique scenarios and to measure its current state. This process would commonly be scheduled every two or three weeks, as to provide the customer with the regular ability to modify the system to their liking. Finally, there is feedback within the team, where in response to the establishment of new requirements from the customer, the team director is required to provide an estimation of time which would be needed to complete implementation.

It has been mentioned that feedback is closely related to both communication and simplicity, and that with the use of various unique and individualised tests can easily be used to strain a system and identify breaks within the code. This then indicates specific tasks and areas for which members of the team should be working on.

❖ *Courage*

Courage can take multiple forms within this development methodology, and can prove very important overall. One instance of this would be that all employees working within the team should feel comfortable within the process of refactoring their work when necessary, and therefore reviewing the system and making modifications for the future integration of code. Additionally, it is important that members of the team possess the courage to identify redundant and obsolete work within the system, and to handle it accordingly. The time and effort which was spent developing this code must be overlooked, and the code removed for beneficial purposes. Finally, persistence is a characteristic which is important within the development team, for there will be instances where a programmer can and will become stuck on a complex problem for long periods of time. Eventually however, and with the right help these problems can easily be overcome and completed, however only if they are persistent.

❖ *Respect*

Respect and self-respect must be present within the workplace when actively developing the necessary software. For starters, in no event should a programmer commit modifications to the system which can break other functions, effectively causing the pre-existing unit-tests to fail, and must generally not hinder their peers in any manner. Additionally, it is important that all programmers present within the team establish a high level of self-respect, as to ensure that they are always striving to complete the best quality of work possible, and therefore seeking the best designs and solutions for development.

Rules

In 1999, the first variations of rules for extreme Programming were initialised by Don Wells, with 29 rules being established around the following categorises; planning, managing, designing, coding, and testing. It could prove beneficial to address however that the planning, managing and designing have made claims that extremely programming does not properly facilitate and support their activities.

An alternative variation of Extreme Programming was then proposed by Ken Auer. He stated that

Chosen Methodology

Having completed an analysis and investigation for a few of the most common methodologies used within software development, I now feel that I am in possession of the knowledge and information needed to make an educated choice regarding the development methodology of this project. Overall, there are a few methodologies that I find relatively interesting, however I have come to the conclusion that I will stick with the standard method of Waterfall. There are a few different supportive reasons for why this decision has been made, all of which generally orientate around the well-defined and organised structure of the waterfall methodology.

As has been previously stated within this document, this method focuses on a structural approach on the development of projects, encouraging and ensuring that systems are pre-planned, designed and created in systematic orders. I deem that this would be relatively beneficial for my approach, for it will help ensure that initial plans and designs can be established to help support the project later within the development cycle.

In addition to this, it also provides a relatively beneficial method which could be used to monitor progress throughout the development system, and help ensure that the overall deadline is met by establishing specific time frames for each area of the project.

Chosen Focus

It is important to address the fact that the school is categorised into three separate areas: “Private Tuition”, “Weekend School”, and “Summer School”. This is problematic for each area contains their own individual features and contents, meaning that the entire system would be relatively extensive. For this reason, I have made the decision to focus my attention towards the “Private Tuition” area. In this area, I can then further focus my attention to the addition, updating, and deletion of various records within the associated tables, and to provide custom displays to filter out desired information.

This is much more practical in many ways, for instance one of the main constraints associated with this project is time, and due to the overall size of the entire system, it would be relatively unrealistic to expect one individual to complete all work within the assigned time frame. By focusing on one aspect, I can effectively improve the quality of the complete area, and ensure that all necessary features are present.

| Section 3 |

| Design |

GANTT chart

Please reference appendix 2

ER Diagram

Please reference appendix 3.

Data Flow Diagram

Please Reference appendix 4.

Normalisation of data

What is Normalisation

The process addressed as Database Normalisation could simply be identified as the organisation of various attributes and tables which would be present within a relational database system. This is done in an attempt to effectively minimise, and thus limit data redundancy within the system. There are a few steps which should be followed to accurately complete this process, however the main concept is that a singular table will be broken down into segments without losing information, while also implementing foreign keys into the initial table, as to reference the primary keys present within the newly defined tables, as to reduce the repetition of information.

There are many reasons for why this is extensively important, one such being that it reduces the amount of time needed to enter new records, making the system more effective. More importantly however, there is the problem that containing multiple instances of the same record can cause conflicts, for while one record may state that the value is a consistent 1, another may state that it is 2. This reduces the dependability of the system, and thus the accuracy of information present.

The primary objective of isolating data, is to create a system in which additions, deletions and general modifications can be made to a specific attribute in a singular table. Then because other tables reference the information present within the selected table, it effectively means that the changes will be expressed throughout the entire system.

Normalisation Process

The normalisation process involves locating and grouping together all entities present within the system, and their associated attributes. Then proceeding to identify and remove any repeating groups of data, and finally provide unique keys for each entity stored within the system. It is important to address the fact that Normalisation uses the concept of Normal Forms, to identify the various steps to Normalise the database. These include:

1st Normal Form

1NF is the first level of normalisation. An entity (table) is in First Normal form if it contains no repeating attributes (fields) or groups of attributes.

To arrange the database information into 1NF, you should reference the following guidelines:

- ❖ Ensure that all repeating groups present within the primary tables are removed, as to guarantee that each record present within the entity are the same length.
- ❖ The groups which have been identified as repeating should then become new entities, linked together by a one-to-many relationship.
- ❖ To create these relationships, a primary key must be present within one entity in the form of a foreign key within another.

2nd Normal Form

An entity is in **2NF** if no attribute (not part of the primary key) is dependent on only part of the primary key. This only applies to entities with concatenated primary keys.

To arrange the database information into 2NF, you should reference the following guidelines:

- ❖ You will need to conduct investigations within the tables to identify any information dependencies. This can be done by individually testing each attribute present, as to check whether it can be uniquely identified by the entire primary key. The only occasion in which these tests would not need to be conducted, is in the event that at least you have established at least one table which requires a concatenated primary key.
- ❖ You will then want to remove all partially dependent attributes, moving them to a new entity.

3rd Normal Form

An entity is in **3NF** if all attributes are entirely dependent on the primary key and not on any attribute that is not part of the primary key.

To arrange the database information into 3NF, you should reference the following guidelines:

- ❖ You will need to initially conduct tests on each attribute, as to ensure they are dependent on the associated primary key.
- ❖ Then proceed to remove all transitive dependencies to a new entity.
- ❖ For reference, a transitive dependency would be identified as an attribute that is dependent on another attribute(s) that are not the primary key.

Six point Plan

To summarise this process, a six point plan has been organised and established. It is as follows:

1NF

- 1) Initially remove any repeating groups which are present within the entity. Then use them to establish new entities.
- 2) Then create a relationship using one of the attributes that are left. In most occasions, this will be the primary key value.

2NF

- 3) Review entities which possess concatenated keys. For in the event that they are not fully dependent on both segments of the primary key, you will want to remove them to a new entity.
- 4) Once again, you will want to then establish relationships between these entities, with the use of one attribute left, that in most occasions will be the primary key.

3NF

- 5) Finally, you will want to review all entities now present within the system. This is for the reason that you will want to identify any attributes that are dependent on another attribute other than the primary key. If this is the case, proceed to once again remove the attribute, and assign it to a new entity.
- 6) Create a relationship between these entities.

ONF

(Student ID, First Name, Other Name(s), Surname, Date of Birth, Address, Town, Post Code, Contact Number, Email Address, Grade ID, Instrument ID, Tuition Fee Received, Teacher ID, First Name, Surname, Address, Town, Post Code, Email Address, Contact Number, Specialisation, Lesson ID, Student ID, Lesson Bundle ID, Purchase Date, Payment Method, Payment Received? Payment Received Date, Bundle Cost (After Discount), Scheduled, Schedule ID, Student ID, Teacher ID, Purchased Lesson ID, Number of weeks, Start Date, Booked Day(s), Booked Time, End Date, Attended, Lesson Bundle ID, Lesson Bundle, Bundle Cost, Multiplier (Discount Rate), Grade ID, Grade, Grade Fees, Room ID, Room Type, Specialisation, Instrument ID, Instrument Type, Instrument Name, Quantity).

1NF

Scheduled Lessons (Schedule ID, s.First Name, s.Other Name(s), s.Surname, s.Date of birth, s.Address, s.Town, s.Post Code, s.Contact Number, s.Email Address, Grade, Grade Fee, Instrument type, Instrument Name, Quantity, t.First Name, t.Surname, t.Address, t.Town, t.Post Code, t.Email Address, t.Contact Number, t.Specialisation, Room type, Specialisation,

Lesson Bundle, Bundle Cost, Multiplier, Number of weeks, Start Date, Booked Day(s), Booked Time, End Date, Attended.)

2NF

Students: (Student ID, First Name, Other Name(s), Surname, Date of birth, Address, Town, Post Code, Contact Number, Email Address, *Grade*, *Grade Fee*, *Instrument type*, *Instrument name*, *Quantity*)

Teachers: (Teacher ID, First Name, Surname, Address, Town, Post Code, Email Address, Contact Number, Specialisation, *Room type*, *Room Specialisation*)

Scheduled Lessons: (Schedule ID, *Lesson Bundle*, *Bundle Cost*, *Multiplier*, Purchase Date, Payment Method, Payment Received, Payment Date, Bundle Cost, Scheduled, Number of weeks, start date, Booked Day(s), Booked Time, End Date, Attended)

3NF

Students: (Student ID, First Name, Other Name(s), Surname, Date of Birth, Address, Town, Post Code, Contact Number, Email Address, Grade ID, Instrument ID, Tuition Fee Received?)

Teachers: (Teacher ID, First Name, Surname, Address, Town, Post Code, Email Address, Contact Number, Specialisation, Room ID)

Purchased Lessons: (Lesson ID, Student ID, Lesson Bundle ID, Purchase Date, Payment Method, Payment Received? Payment Received Date, Bundle Cost (After Discount), Scheduled)

Schedule: (Schedule ID, Student ID, Teacher ID, Lesson Bundle ID, Number of weeks, Start Date, Booked Day(s), Booked Time, End Date, Attended)

Lesson Bundles: (Lesson Bundle ID, Lesson Bundle, Bundle Cost, Multiplier (Discount Rate))

Grades: (Grade ID, Grade, Grade Fees)

Rooms: (Room ID, Room Type, Specialisation)

Instruments: (Instrument ID, Instrument Type, Instrument Name, Quantity)

Inputs

Students: Student ID, First Name, Other Name(s), Surname, Date Of Birth, Address, Town, Post Code, Contact Number, Email Address, Grade ID, Instrument ID, Tuition Fee Received.

Teachers: Teacher ID, First Name, Surname, Address, Town, Post Code, Email Address, Contact Number, Specialisation, Room ID.

Instruments: Instrument ID, Type, Name, Quantity.

Rooms: Room ID, Type, Specialisation.

Grades: Grade ID, Level, Fee.

Lesson Bundles: Lesson Bundle ID, Lesson Bundle, Bundle Cost, Multiplier (Discount).

Scheduled Lessons: Schedule ID, Student ID, Teacher ID, Purchased Lesson ID, Number of Weeks, Start Date, Booked Day(s), Booked Time, End Date.

Purchased Lessons: Purchased Lessons ID, Student ID, Lesson Bundle ID, Purchase Date, Payment Method, Payment Received?, Payment Received Date, Bundle Cost (After Discount), Scheduled.

Reallocated Students: Student ID, First Name, Other Name(s), Surname, Date Of Birth, Address, Town, Post Code, Contact Number, Email Address, Grade ID, Instrument ID, Tuition Fee Received.

Archived Scheduled: Schedule ID, Student ID, Teacher ID, Purchased Lesson ID, Number of Weeks, Start Date, Booked Day(s), Booked Time, End Date.

Lesson Dates (Scheduled Lessons elaboration): Lesson Dates ID, Scheduled_Lessons ID, Date, Attended.

Archived Lesson Dates: LessonDates_Archive ID, LessonDates ID, Scheduled Lessons ID, Date, Attended.

Data Dictionary

Student Table

Attribute	Data type	Field size	Validation	Purpose	Comments
Student ID	Int	-	<p>This field has been assigned the following properties of: Identity Specification, (Is Identity = true), and NOT NULL.</p> <p>This simply indicates that the field has been designated as the student table's primary key. In addition to this, it will also assign values automatically, in an incremental fashion, ensuring that no duplicated values are ever present.</p> <p>Establishing this field as NOT NULL, implies that a value must be present within this field for all records.</p>	<p>This field will be situated as the primary key for the student table, being used to uniquely identify all associated records.</p>	<p>This field will be used to uniquely identify and thus reference student records.</p>
First_Name	Nvarchar (50)	50	First Name values will be entered through the use of a text box. These values will then be analysed for the presence of invalid characters, those being any that are non-alphabetical.	As the name of this field may imply, it will simply be used to store the Forename for each student.	N/A

			<p>In addition to this, the textbox has been assigned a max character limit of '50'.</p> <p>This field has also been established as NOT NULL, which implies that a value must be present within this field for all records.</p>		
Other_Name(s)	Nvarchar (50)	50	<p>Other Name values will be entered through the use of a text box. These values will then be analysed for the presence of invalid characters, those being any that are non-alphabetical.</p> <p>In addition to this, the textbox has been assigned a max character limit of '50'.</p> <p>This field has also been established as NOT NULL, which implies that a value must be present within this field for all records.</p>	<p>This field will be used to store any additional names which a student may possess.</p>	N/A
Surname	Nvarchar (50)	50	<p>Surname values will be entered through the use of a text box. These values will then be analysed for the presence of invalid characters, those</p>	<p>This field will be used to store the surname for each individual student.</p>	N/A

			<p>being any that are non-alphabetical.</p> <p>In addition to this, the textbox has been assigned a max character limit of '50'.</p> <p>This field has also been established as NOT NULL, which implies that a value must be present within this field for all records.</p>		
Date_of_Birth	Date	11	<p>To validate this field, the user will select a value present within a calendar to represent the desired date. This will ensure that the correct format is used. In addition to this, upon attempting to submit a DOB value, it will be checked to ensure that it is at least 4-5 years prior to the current date.</p> <p>This should in theory reduce the overall chance of a user assigning a student who was born yesterday or tomorrow.</p> <p>In addition to this, this textbox has also been assigned a max length value of '11'.</p> <p>Finally, it should be noted that this</p>	<p>This field is used to store the date for which each student was born.</p>	<p>Interestingly, the validation will occur in relation to a textbox object, however it will be locked from the user, preventing them from making direct modifications to the data.</p> <p>Instead, in company of the textbox, a button will be present. When selected this should cause the mentioned calendar to appear, providing the user with the ability to select a value and indirectly present it within the textbox.</p>

			<p>field has been established as NOT NULL, which simply implies that a value must be present for each record.</p>		
Address	Nvarchar (50)	50	<p>The data for this field would be collected within a textbox object, which would then be segmented into two sections. This would be done by searching the provided string for the presence of a 'Space' value, and then identifying the 'first = house number and 'last = street name' segments of the string.</p> <p>Each of these separate sections could then be passed through forms of validation, as to ensure that no invalid characters are present within either. Extensive measures have also been taken to provide specific user feedback, representing the fact that in most cases a direct issue should be stated.</p> <p>In addition to this, this textbox has been assigned a max length value of '50'.</p> <p>Finally, it should be noted that this</p>	<p>To store the students address, that may be needed for the delivery of special documents and materials.</p>	<p>The validation used in relations to this field could be deemed relatively complex. This is due to the fact that an address value will contain both numerical and alphabetical characters. Yet this would commonly be done within two sections, one of which should only contain numbers, and the other only letters.</p>

			<p>field has been established as NOT NULL, which simply implies that a value must be present for each record.</p>		
Town	Nvarchar (50)	50	<p>This field is collected within a textbox, and contains simple validation as to ensure that no invalid characters are present within the submitted value. For instance, a town name should not contain numerical, symbolic, or punctual characters.</p> <p>In addition to this, this textbox has been assigned a max length value of '50'.</p> <p>Finally, it should be noted that this field has been established as NOT NULL, which simply implies that a value must be present for each record.</p>	<p>The town field will be used to expand upon the address field, storing the current town the student lives within.</p>	N/A
PostCode	Nvarchar (7)	50	<p>The postcode value is collected within a textbox object, and would initially be dissected into three new variables. The first of which consists of the first two letters, which must be some form of "BT". The next has been identified as the numbers, and finally the last</p>	<p>The post code field will finalise the students address, storing a post code which represents the location of the students current accommodations.</p>	<p>The Postcode could prove relatively problematic when in relations to validation, due to the specific format that is enforced. Once which I was able to break down into "BT"-“Numbers”-</p>

			<p>section would letters.</p> <p>These final two sections would be run through various checks as to ensure that there are no invalid characters present within each.</p> <p>It would also be beneficial to address the fact that this textbox has been assigned a maxlength limit of '7'.</p> <p>Finally, it should be noted that this field has been established as NOT NULL, which simply implies that a value must be present for each record.</p>		"Letters".
Contact_Number	Int	11	<p>To validate the contact Number, the decision was made to simply test the value for the presence of any non-numeric values. This was because an alphabetical character or symbol provides no benefit to this field. In addition to this, validation is used to ensure that the information entered into this field does not exceed the character limit of 11.</p>	<p>This field is responsible for storing the students contact Number, most commonly their mobile. It is also important to acknowledge that for students under the age of 18, the contact number for their parent or guardian will be added.</p> <p>This provides the school with the ability to communicate important</p>	N/A

				information, and establish communication with their students.	
Email_Address	Nvarchar (75)	75	<p>New values for this field will be collected within a textbox. Then upon submission, a simple check will occur as to ensure that the '@' character is present.</p> <p>In addition to this, this textbox has been assigned a max length value of '75'.</p> <p>Finally, it should be noted that this field has been established as NOT NULL, which simply implies that a value must be present for each record.</p>	<p>This field is responsible for the storage of a student's Email address.</p> <p>This provides members of staff with the ability to communicate with students and transfer important pieces of documentation.</p>	N/A
GradeID	Int	-	<p>The GradeID value will be entered through the use of a combo box, containing information regarding all associated IDs present within the system.</p> <p>This will ensure that the user selects a valid value, one which directly correlates with the grades available</p>	<p>This field will operate as the Grade foreign key, effectively connecting the grade table to the students table.</p> <p>This is accomplished by entering a value which corresponds to that of a</p>	N/A

			<p>within the system.</p> <p>Finally, it should be noted that this field has been established as NOT NULL, which simply implies that a value must be present for each record.</p>	<p>key field within the grade table.</p>	
Instrument ID	Int	-	<p>Similarly, the Instrument ID will also be collected through the use of a combo box, one which will detail information in correspondence to the various Instruments present within the system.</p> <p>Finally, it should be noted that this field has been established as NOT NULL, which simply implies that a value must be present for each record.</p>	<p>Similarly, the instrument ID field will operate as the foreign key, this time to effectively connect the instrument table to the students table.</p>	N/A
Tuition_Fee_Received	Bit	1	<p>This value will likely be entered into the system from a checkbox object. This simply provides users with the ability to either select it, or leave it unselected. This effectively represents two possibilities, "True or False".</p>	<p>This will simply determine whether or not the student in question has paid their tuition fee.</p>	<p>A bit value uses the following digits to express a true/false expression, where 0 = False, and 1 = True.</p>

Teacher Table

Attribute	Data type	Field size	Validation	Purpose	Comments
TeacherID	Int	1	This field will automatically assign each new record a unique referential value. The user has no direct control or input in relations to this field, therefore it does not require validation.	The teacher ID is used to uniquely identify each teacher within the table.	The teacher ID is the primary key for the teacher table, and uses the following properties to complete said purpose: Is Identify = Yes, Increment = 1. This means that the field will automatically assign each new record a value, one which is an increment of that previously used.
First_Name	Nvarchar (50)	50	First Name values will be collected through the use of a textbox, and would then be validated for the presence of any characters deemed invalid. This may include numerical, symbolic and punctual characters. In addition to this, this textbox has been assigned a max length value of '50'. Finally, it should be noted that this field has been established as NOT	The first name field is used to store the forename for the teacher, and would be used for identification and general reference.	N/A

			NULL , which simply implies that a value must be present for each record.		
Surname	Nvarchar (50)	50	<p>Surname values will be collected through the use of a textbox, and would then be validated for the presence of any characters deemed invalid. This may include numerical, symbolic and punctual characters.</p> <p>In addition to this, this textbox has been assigned a max length value of '50'.</p> <p>Finally, it should be noted that this field has been established as NOT NULL, which simply implies that a value must be present for each record.</p>	<p>Similarly, this field would be used to store the teachers surname, and would be used to further identify the teacher.</p>	N/A
Address	Nchar (60)	60	The data for this field would be collected within a textbox object, which would then be segmented into two sections. This would be done by searching the provided string for the presence of a 'Space' value, and then identifying the 'first = house number and 'last = street name' segments of the string.	This field is used to store the address information for each teacher.	N/A

			<p>Each of these separate sections could then be passed through forms of validation, as to ensure that no invalid characters are present within either. Extensive measures have also been taken to provide specific user feedback, representing the fact that in most cases a direct issue should be stated.</p> <p>In addition to this, this textbox has been assigned a max length value of '50'.</p> <p>Finally, it should be noted that this field has been established as NOT NULL, which simply implies that a value must be present for each record.</p>		
Town	Nchar (60)	60	<p>This field is collected within a textbox, and contains simple validation as to ensure that no invalid characters are present within the submitted value. For instance, a town name should not contain numerical, symbolic, or punctual characters.</p> <p>In addition to this, this textbox has</p>	<p>This an expansion upon the address field, and would store the town for which the teachers home is located within or around.</p>	N/A

			<p>been assigned a max length value of '50'.</p> <p>Finally, it should be noted that this field has been established as NOT NULL, which simply implies that a value must be present for each record.</p>		
PostCode	Nchar (7)	7	<p>The postcode value is collected within a textbox object, and would initially be dissected into three new variables. The first of which consists of the first two letters, which must be some form of "BT". The next has been identified as the numbers, and finally the last section would letters.</p> <p>These final two sections would be run through various checks as to ensure that there are no invalid characters present within each.</p> <p>It would also be beneficial to address the fact that this textbox has been assigned a maxlen limit of '7'.</p> <p>Finally, it should be noted that this field has been established as NOT</p>	<p>This field is the final extension to the address field, storing the postcode which directly correlates to the location of the teacher's home.</p>	N/A

			NULL, which simply implies that a value must be present for each record.		
Email_Address	Nchar (60)	60	<p>New values for this field will be collected within a textbox. Then upon submission, a simple check will occur as to ensure that the '@' character is present.</p> <p>In addition to this, this textbox has been assigned a max length value of '75'.</p> <p>Finally, it should be noted that this field has been established as NOT NULL, which simply implies that a value must be present for each record.</p>	<p>This field is used to detail Email address for the teachers present within the school – be that personal, or one assigned by the school.</p> <p>These values would be used to contact, or distribute important documents or information through the use of the internet.</p>	N/A
Contact_Number	Nchar (11)	11	<p>New Values will be entered through the use of a textbox. Therefore to validate the information, the decision was made to restrict the objects max length to 11, and to ensure that the value is composed of numerical characters solely.</p> <p>Finally, it should be noted that this field has been established as NOT</p>	N/A	N/A

			NULL, which simply implies that a value must be present for each record.		
Specialisation	Nchar (40)	40	<p>New values will be entered into this field through the use of a combo box, one which will store a list of all instrumental groups, with a few additions.</p> <p>Finally, it should be noted that this field has been established as NOT NULL, which simply implies that a value must be present for each record.</p>	<p>This field is used to identify the instrumental specialisation for each teacher. This is important, for it ensure that they are assigned to classes where they would be most efficient.</p>	N/A
RoomID	int	-	<p>Values for this field will also be inputted into the system through the use of a combo box, this one detailing information in association to all rooms documented within the system.</p> <p>Finally, it should be noted that this field has been established as NOT NULL, which simply implies that a value must be present for each record.</p>	<p>This field is used to effectively link the room table, and to therefore identify the room for which the designated teacher has been assigned to.</p>	N/A

Scheduled Lessons Table

Attribute	Data type	Field size	Validation	Purpose	Comments
ScheduleID	Int	-	This field should automatically assign each new record a unique referential value. The user has no direct control or input in relations to this field, therefore it does not require validation.	The Schedule ID is used to uniquely identify each teacher within the table.	The Schedule ID is the primary key for the teacher table, and uses the following properties to complete said purpose: Is Identify = Yes, Increment = 1. This means that the field will automatically assign each new record a value, one which is an increment of that previously used.
StudentID	Int	-	New values will be entered into this field through the use of a combo box, one which will store values associated and representative of all student records present within the system. Finally, it should be noted that this field has been established as NOT NULL, which simply implies that a	This field is used to assign a specific student to the scheduled lesson record.	N/A

			<p>value must be present for each record.</p>		
TeacherID	Int	-	<p>New values will be entered into this field through the use of a combo box, one which will store values associated and representative of all Teachers records present within the system.</p> <p>Finally, it should be noted that this field has been established as NOT NULL, which simply implies that a value must be present for each record.</p>	<p>This field is used to assign a specific Teacher to the scheduled lesson record.</p>	N/A
PurchasedLessonID	Int	-	<p>New values will be entered into this field through the use of a combo box, one which will store values associated and representative of all Purchased Lesson records present within the system.</p> <p>Finally, it should be noted that this field has been established as NOT NULL, which simply implies that a value must be present for each record.</p>	<p>This field is used to assign a specific Purchased Lesson to the scheduled lesson record.</p>	N/A

Number_Of_Weeks	Nchar(20)	20	<p>New values will be entered into this field through the use of a combo box present within the associated form. The combo box in question will store values such as: (“5 Weeks, 10 Weeks, 15 Weeks” etc.)</p> <p>In addition to this, the combo box has been modified to ensure that the user does not possess the ability to make custom modifications to the data.</p> <p>Finally, it should be noted that this field has been established as NOT NULL, which simply implies that a value must be present for each record.</p>	N/A	N/A
Start_Date	Date	11	<p>New values will be entered into this field through the use of an associated calendar object. This is beneficial for it effectively validates the new information, for a default format will be assigned, and it ensure that the entered value is a valid date overall.</p>	<p>To access this calendar, a button will be present within the associated form.</p>	N/A

			<p>Finally, it should be noted that this field has been established as NOT NULL, which simply implies that a value must be present for each record.</p>		
Booked_Day(s)	Nchar (100)	100	<p>New values will be entered through the use of multiple check boxes. A custom menu will display multiple Check boxes, one representing each day of the week (Bar the weekends).</p> <p>In addition to this, this information will be validated by reviewing the number of selected days. The user should be restricted from selecting more than 2 days.</p> <p>Finally, it should be noted that this field has been established as NOT NULL, which simply implies that a value must be present for each record.</p>	<p>This menu can be accessed through the use of an associated button.</p>	N/A
Booked_Time	Time(7)	7	New values for this field will be entered through the use of a combo box present within a form. The combo box in question will	N/A	N/A

			<p>contain time values in 30 minute intervals, ranging from 13:00 – 18:30.</p> <p>Finally, it should be noted that this field has been established as NOT NULL, which simply implies that a value must be present for each record.</p>		
End_Date	Nchar(10)	10	<p>New values will be entered into this field through the use of an associated calendar object. This is beneficial for it effectively validates the new information, for a default format will be assigned, and it ensure that the entered value is a valid date overall.</p> <p>Finally, it should be noted that this field has been established as NOT NULL, which simply implies that a value must be present for each record.</p>	<p>To access this calendar, a button will be present within the associated form.</p>	N/A

Lesson Purchase Table

Attribute	Data type	Field size	Validation	Purpose	Comments
LessonPurchasedID	Int	-	This field should automatically assign each new record a unique referential value. The user has no direct control or input in relations to this field, therefore it does not require validation.	The LessonPurchased ID is used to uniquely identify each teacher within the table.	The LessonPurchased ID is the primary key for the teacher table, and uses the following properties to complete said purpose: Is Identify = Yes, Increment = 1. This means that the field will automatically assign each new record a value, one which is an increment of that previously used.
StudentID	Int	-	New values will be entered into this field through the use of a combo box, one which will store values associated and representative of all student records present within the system. Finally, it should be noted that this field has been established as NOT NULL, which simply implies that a value must be present for each record.	This field is used to assign a specific student to the scheduled lesson record.	N/A

LessonBundleID	Int	-	<p>New values will be entered into this field through the use of a combo box, one which will store values associated and representative of all Lesson Bundle records present within the system.</p> <p>Finally, it should be noted that this field has been established as NOT NULL, which simply implies that a value must be present for each record.</p>		
Purchased_Date	Date	11	<p>New values will be entered into this field through the use of a calendar present within a form, which implies that the user will have no direct control of the values input format. This is beneficial for it effectively ensures that entered values take a beneficial and valid format, and that they represent acceptable dates.</p> <p>Finally, it should be noted that this field has been established as NOT NULL, which simply implies that a value must be present for each</p>	<p>This calendar will be accessible through the use of a button present within the same form.</p>	N/A

			record.		
Payment_Method	Nchar(25)	25	<p>New values will be entered into this field through the use of a combo box, one which will store payment method values such as: "Cash, Card, Cheque" etc.</p> <p>Finally, it should be noted that this field has been established as NOT NULL, which simply implies that a value must be present for each record.</p>	<p>This is used to identify the student selected payment method.</p>	N/A
Payment_Received	Bit	1	<p>New values will be entered into this field with the use of a Check box. This is the potentially the most practical method to use when collecting and transferring BIT data, for both represent one of two values True(1) or False(0).</p> <p>Finally, it should be noted that this field has been established as NOT NULL, which simply implies that a value must be present for each record.</p>	<p>Used to represent the fact of whether or not the student has paid for the lesson.</p>	N/A

Payment_Received Date	Nchar(10)	10	<p>New values will be entered into this field through the use of a calendar present within a form, which implies that the user will have no direct control of the values input format. This is beneficial for it effectively ensures that entered values take a beneficial and valid format, and that they represent acceptable dates.</p> <p>Finally, it should be noted that this field has been established as NOT NULL, which simply implies that a value must be present for each record.</p>	<p>This calendar will be accessible through the use of a button present within the same form.</p>	N/A
Bundle_Cost(After_Discount)	Decimal(18,0)	18	<p>New values entered into this field will be automatic and therefore restrict the use from having direct control.</p>	<p>Bundle Cost is calculated in association to multiple variables throughout the system. This means that the value will be assigned automatically, once the value in question has been calculated.</p>	N/A

Lesson Bundle Table

Attribute	Data type	Field size	Validation	Purpose	Additional Information
LessonBundleID	Int	-	This field should automatically assign each new record a unique referential value. The user has no direct control or input in relations to this field, therefore it does not require validation.	The LessonBundle ID is used to uniquely identify each teacher within the table.	The LessonBundle ID is the primary key for the teacher table, and uses the following properties to complete said purpose: Is Identify = Yes, Increment = 1. This means that the field will automatically assign each new record a value, one which is an increment of that previously used.
Lesson Bundle	Nchar(40)	40	New values will be entered into this field through the use of a simple textbox. This will provide users with the ability to enter the desired value, which will then be validated to ensure that only alphabetical characters are present. In addition to this, the decision was made to establish a max length limit of 50 for the textbox component.	This is simply the name of the lesson bundle, and represents the number of lessons associated.	N/A

			<p>Finally, it should be noted that this field has been established as NOT NULL, which simply implies that a value must be present for each record.</p>		
Bundle Cost		-	<p>New values will be entered into this field through the use of a simple textbox. This will provide users with the ability to add the desired value. The system will then proceed to automatically validate the entered value, as to ensure that only numerical characters are present.</p> <p>Finally, it should be noted that this field has been established as NOT NULL, which simply implies that a value must be present for each record.</p>	<p>This value represents the bundle cost for each lesson bundle.</p>	N/A
Multiplier (Discount Rate)	Float	-	<p>New values will be entered into this field through the use of a simple textbox. This will provide users with the ability to add the desired value. The system will then proceed to automatically validate</p>	<p>This value represents the multiplier value which would be used to represent the discount.</p>	N/A

		<p>the entered value, as to ensure that only numerical characters are present.</p> <p>Finally, it should be noted that this field has been established as NOT NULL, which simply implies that a value must be present for each record.</p>		
--	--	--	--	--

Room Table

Attribute	Data type	Field size	Validation	Purpose	Additional Information
RoomID	Int	-	This field should automatically assign each new record a unique referential value. The user has no direct control or input in relations to this field, therefore it does not require validation.	The Room ID is used to uniquely identify each teacher within the table.	The Room ID is the primary key for the teacher table, and uses the following properties to complete said purpose: Is Identify = Yes, Increment = 1. This means that the field will automatically assign each new record a value, one which is an increment of that previously used.
Room_Type	Nchar(25)	25	New values will be entered into this field through the use of a simple textbox component. This provides users with the ability to easily type the desired value. Upon confirming the value, the system will then proceed to automatically validate the new information, as to ensure that there are only alphabetical characters present. Finally, it should be noted that this field has been established as NOT	This is used to represent the overall function of the selected room.	N/A

			<p>NULL, which simply implies that a value must be present for each record.</p>		
Specialisation	Nchar(50)	50	<p>New values will be entered into this field through the use of a combo box, one which details a range of instrument groups and specialisations.</p> <p>It is also beneficial to address the fact that properties have been modified as to ensure that the user does not possess the ability to add or customise values present within the combo box.</p> <p>Finally, it should be noted that this field has been established as NOT NULL, which simply implies that a value must be present for each record.</p>	<p>This is used to represent the purpose of the room, and thus the lessons which would be held.</p> <p>N/A</p>	

Instrument Table

Attribute	Data type	Field size	Validation	Purpose	Additional Information
Instrument ID	Int	-	This field should be automatic, and thus does not require validation.	The purpose of this field is to provide each instrument record with an individualised identifier. This ensures that information will not overlap, and simplifies the process of accessing the record within future dates.	The Room ID is the primary key for the teacher table, and uses the following properties to complete said purpose: Is Identify = Yes, Increment = 1. This means that the field will automatically assign each new record a value, one which is an increment of that previously used.
Instrument Type	Nchar(50)	50	This field must only contain alphabetical values, for numerical values express nothing of importance in this field. In addition to this, any values entered into this field must not exceed the specified character limit of 50.	This field is used to associate the instrument with one of the five primary instrument groups; "String", "Percussion", "Brass", "Keyboard", "Wood Wind", in addition to thus, "Vocal" will also be accepted.	It is extremely likely that a drop-down menu will be used to add a value into this field, due to the fact that there are only six values which would be desirable within this field. This also simplifies the validation process, for a simple check could be done to ensure that the selected value is present within the list.

Instrument Name	Nchar(50)	50	Similarly, this field must also only contain alphabetical values, for the same reasoning.	This field is used to store the name associated with the instrument referenced within the record.	N/A
Quantity	Int	-	This field must only contain numerical values, for alphabetical characters and symbols will express nothing of importance within this field. In addition to this, a range of “>0 AND <15” will be established.	This field is used to identify the total quantity of each instrument currently present within the schools storage.	N/A

Grade Table

Attribute	Data type	Field size	Validation	Purpose	Additional Information
Grade ID	Int	-	The user will not be allowed to directly access or modify this field.	This field is used to assign each record a unique identifier which can be accessed throughout the system. This ensures that the data does not overlap and become ineffective.	Is Identity – Identify Increment = 1. This simply means that this field has been designated as the primary key for the grade table. All records will be possess a unique value within this field.
GradeLevel	Nchar(25)	25	The grade level cannot contain a numerical value, and must not exceed the field size of 25.	This field is used to store the overall name assigned to the grade in question.	N/A
Grade Fee	Int	-	The grade fee is opposite the previous field, it must only contain numerical values (Cost symbols may be added by the system automatically). In addition to this, entered values must fall within the following range: ">0 AND <50".	This field is then used to store the cost associated with the grade, a value which will be necessary within the calculation process of total costs.	N/A

Achieved Lessons Table

Attribute	Data type	Field size	Validation	Purpose	Additional Information
ScheduleID	Int	-	N/A	To store archived information.	N/A
StudentID	Int	-	N/A	To store archived information.	N/A
TeacherID	Int	-	N/A	To store archived information.	N/A
LessonBundleID	Int	-	N/A	To store archived information.	N/A
Number_Of_Weeks	Nchar(20)	20	N/A	To store archived information.	N/A
Start_Date	Date	11	N/A	To store archived information.	N/A
Booked_Day(s)	Nchar (100)	100	N/A	To store archived information.	N/A
Booked_Time	Time(7)	7	N/A	To store archived information.	N/A
End_Date	Nchar(10)	10	N/A	To store archived information.	N/A

Reallocated Student table

Attribute	Data type	Field size	Validation	Purpose	Additional Information
StudentID	Int	-	N/A	To store archived information.	N/A
First_Name	Nvarchar(50)	50	N/A	To store archived information.	N/A
Other_Name(s)	Nvarchar(50)	50	N/A	To store archived information.	N/A
Surname	Nvarchar(50)	50	N/A	To store archived information.	N/A
Date_of_Birth	Date	11	N/A	To store archived information.	N/A
Address	Nvarchar(50)	50	N/A	To store archived information.	N/A
Town	Nvarchar(50)	50	N/A	To store archived information.	N/A
PostCode	Nvarchar(8)	8	N/A	To store archived information.	N/A
Contact_Number	Int	11	N/A	To store archived information.	N/A

Email_Address	Nvarchar(75)	75	N/A	To store archived information.	N/A
GradeID	Int	-	N/A	To store archived information.	N/A
InstrumentID	Int	-	N/A	To store archived information.	N/A
Tuition_Fee_Received	Bit	1	N/A	To store archived information.	N/A

Lesson Dates

Attribute	Data type	Field size	Validation	Purpose	Additional Information
LessonDateID	Int	-	This field should automatically assign each new record a unique referential value. The user has not direct control or input in relations to this field, therefore it does not require validation.	The LessonDate ID is used to uniquely identify each teacher within the table.	The LessonDate ID is the primary key for the teacher table, and uses the following properties to complete said purpose: Is Identify = Yes, Increment = 1. This means that the field will automatically assign each new record a value, one which is an increment of that previously used.
ScheduleID	Int	-	New information will be automatically entered into this field by the system when a new scheduled lesson is added.	The system will use the system information to calculate the various lesson dates, and then proceed to add the new information.	N/A
Date	DateTime	11	New information will be automatically entered into this field by the system when a new scheduled lesson is added.	The system will use the system information to calculate the various lesson dates, and then proceed to add the new information.	N/A

Attended?	Bit	1	New information will be automatically entered into this field by the system when a new scheduled lesson is added.	The system will use the system information to calculate the various lesson dates, and then proceed to add the new information.	N/A
------------------	-----	---	---	--	-----

Processes

- ❖ Add New Record:
 - Add New Student Record,
 - Add New Teacher Record,
 - Add New Instrument Record,
 - Add New Grade Record,
 - Add New Lesson Bundle Record,
 - Add New Room Record,
 - Add New Purchased Lesson Record,
 - Add New Scheduled Lesson Record,
- ❖ Update Record:
 - Update Student Record,
 - Update Teacher Record,
 - Update instrument Record,
 - Update Grade Record,
 - Update Lesson Bundle Record,
 - Update Room Record,
 - Update Purchased Lesson Record,
 - Update Scheduled Lesson Record,
- ❖ Delete
 - Delete Student Record,
 - Delete Teacher Record,
 - Delete Instrument Record,
 - Delete Grade Record,
 - Delete Lesson Bundle Record,
 - Delete Room Record,
 - Delete Purchased Lesson Record,
 - Delete Scheduled Lesson Record,
- ❖ Calculate and Update Bundle Cost,
- ❖ Calculate and Update End Date,
- ❖ Filter Scheduled Lessons in calendar,
- ❖ Filter Scheduled Lessons in Upcoming Lessons,

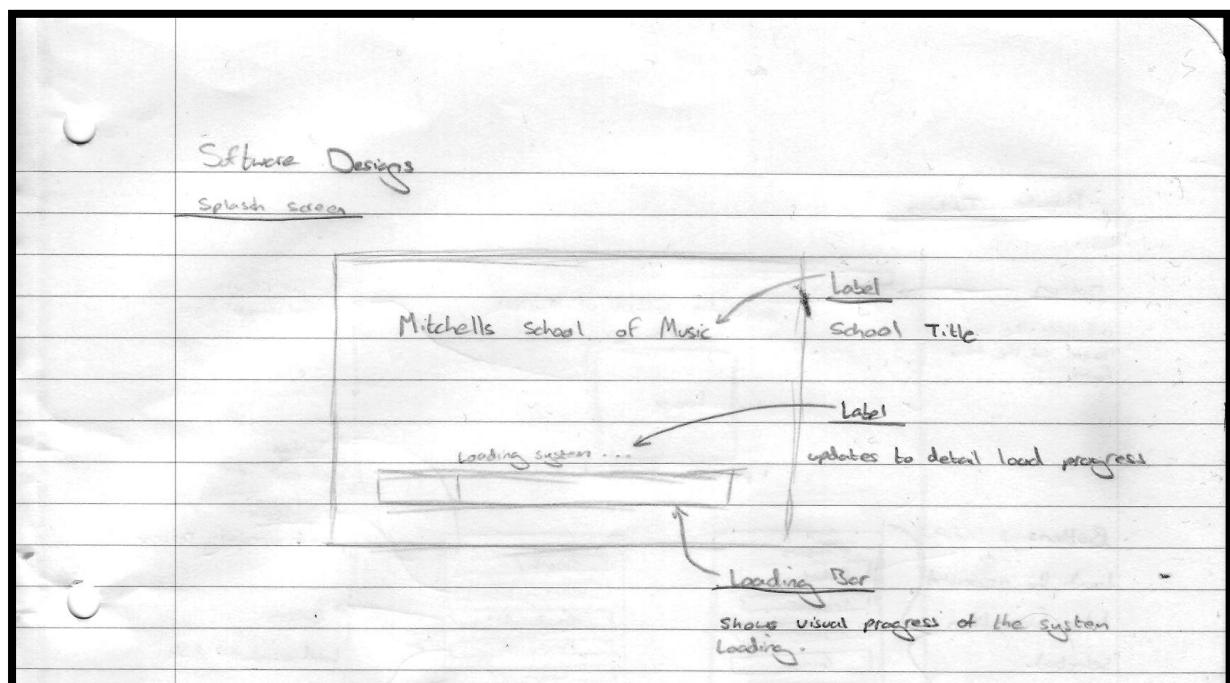
Outputs

❖ Display

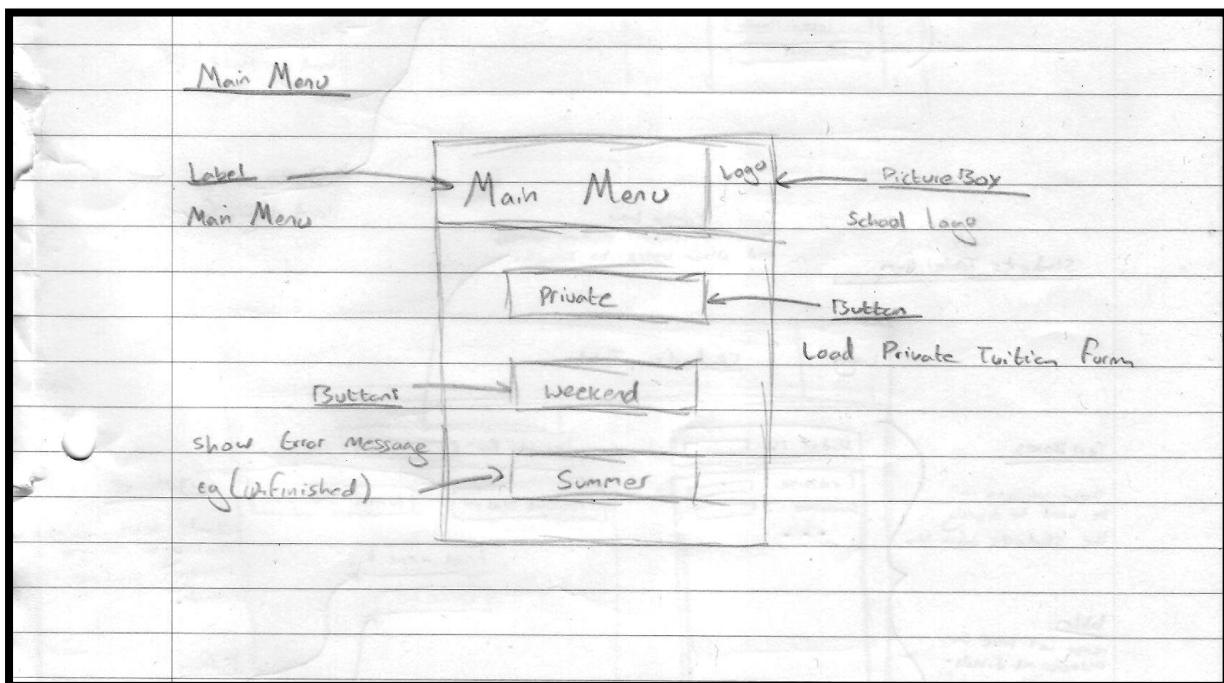
- **Display Student Details,**
- **Display Teacher Details,**
- **Display Instrument Details,**
- **Display Grade Details,**
- **Display Lesson Bundle Details,**
- **Display Room Details,**
- **Display Purchased Lesson Details,**
- **Display Scheduled Lesson Details,**

Draft Story Boards

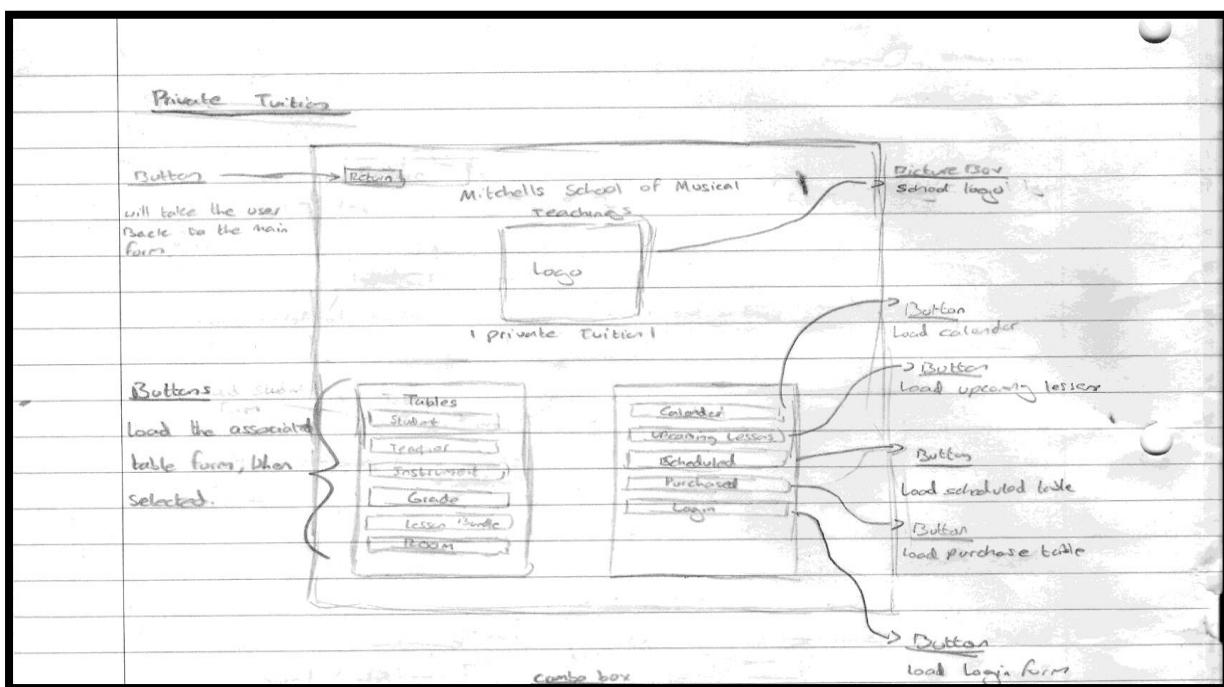
1) Splash Screen



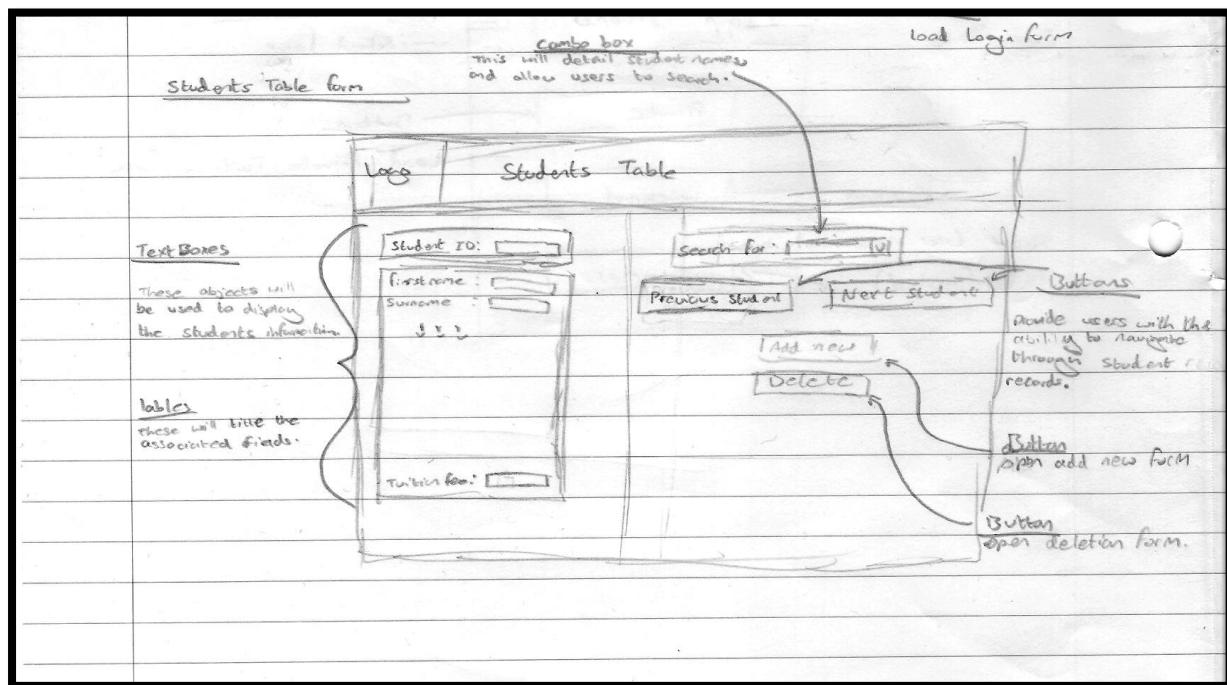
2) Selection Menu



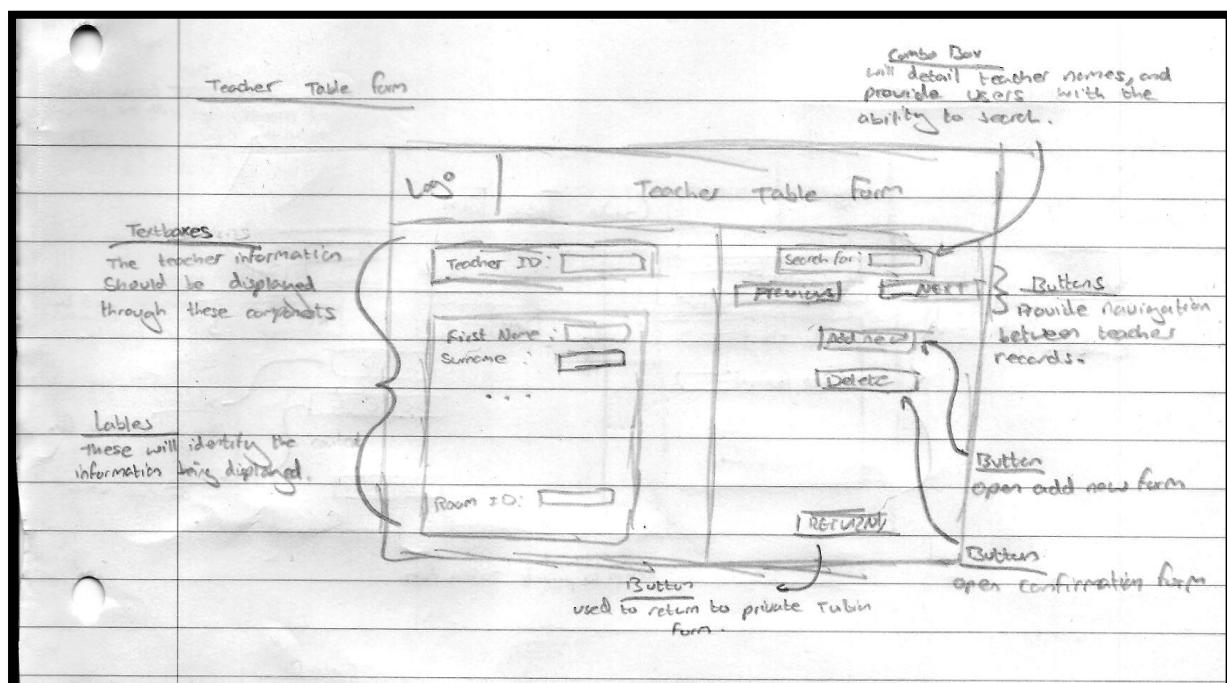
3) Private Tuition Menu



4) Students List



5) Teachers List



6) Instrument List

Instrument Table Form

Text Boxes
These objects are used to display the instrument information.

Combo Box
Stores instrument name. Provides users with the ability to search.

Buttons
Provide the user with the ability to effectively navigate between teacher records.

Button
opens the add field form

Button
opens the confirmation form

Button
used to return to private Tuition form.

7) Grade List

Lesson Bundle List

Text Boxes
These will be used to & display the information.

Combo Box
This will store the title for each lesson bundle present within the system. This will provide the user with the ability to search.

Buttons
These buttons would be used to navigate bundle records.

Button
opens Add Field form

Button
opens confirmation form

Button
used to return to private Tuition form.

8) Lesson Bundle List

Grade Table Form

Text Boxes
These will be used to display the Grade information.

Combo Box
This will store all Grade Names and provide users with the ability to search.

Buttons
these are used to navigate between Grade records.

Buttons
→ Button open the Add field form

Buttons
→ Button open the Confirmation form.

Buttons
used to return to the private Tuition Form.

9) Schedule Calendar [Date Selection]

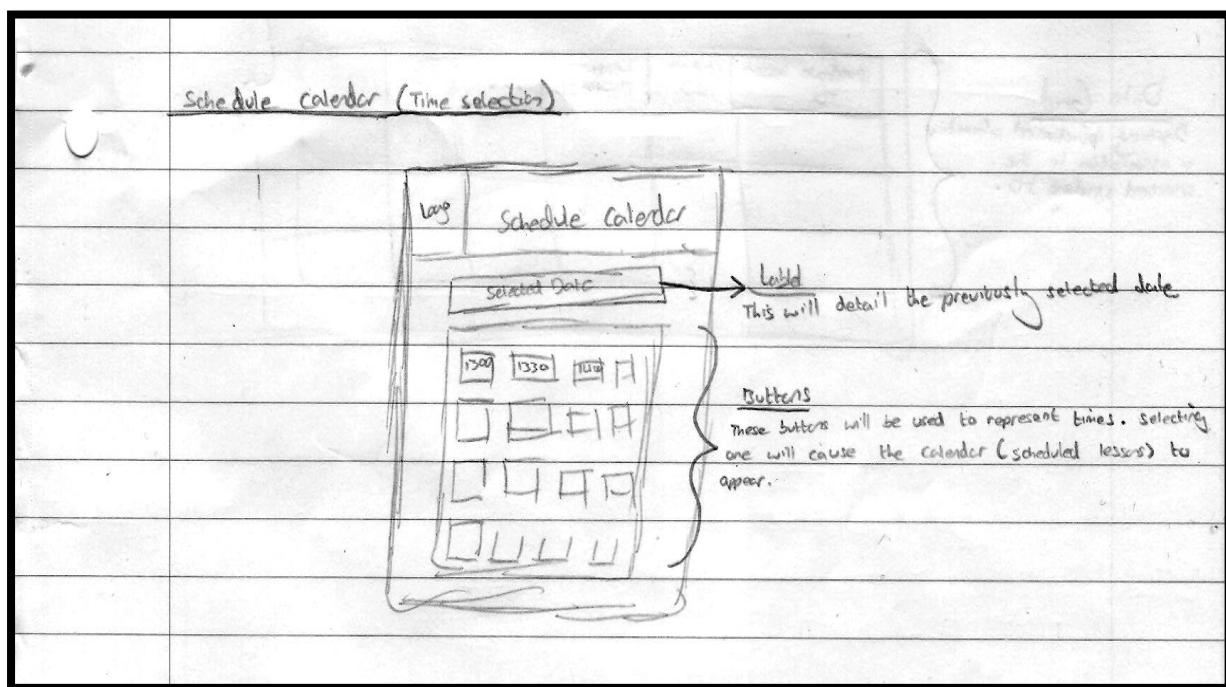
Schedule Calender (Date Selection)

Buttons
These buttons would be used to select the desired month.

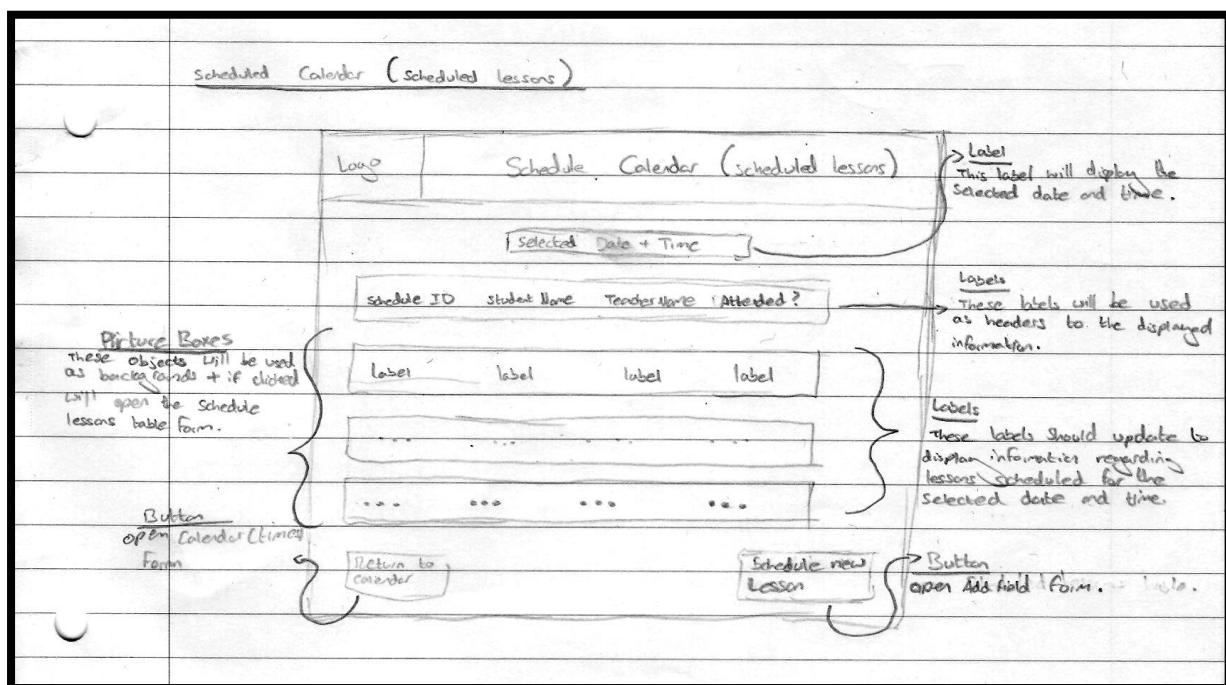
Buttons
These buttons would be used to select the desired date. selecting will cause calendar selection to appear.

Buttons
used to return to the private tuition form.

10) Schedule Calendar [Time Selection]



11) Schedule Calendar [Scheduled Lessons]



12) Schedule Lessons List

Purchased Lessons Table Form

Logo	Purchased Lessons Table																						
List Boxes These List Boxes are used to display basic information in association to the selected student and purchased lesson.	Student ID: <input type="text"/> First Name: <input type="text"/> Other Names: <input type="text"/> Contact Number: <input type="text"/>	Lesson Bundle ID: <input type="text"/> Bundle Title: <input type="text"/> Bundle Fee: <input type="text"/>	Search Box: <input type="text"/> <input type="button" value="SEARCH"/> <input type="button" value="REFRESH"/> <input type="button" value="ADD"/> <input type="button" value="DELETE"/> <input type="button" value="CONFIRMATION"/>																				
Data Grid Displays purchased information in association to the selected student ID.	<table border="1"> <thead> <tr> <th>Purchased Lesson ID</th> <th>Student ID</th> <th>Lesson Bundle ID</th> <th>...</th> </tr> </thead> <tbody> <tr><td> </td><td> </td><td> </td><td> </td></tr> </tbody> </table>			Purchased Lesson ID	Student ID	Lesson Bundle ID	...																
Purchased Lesson ID	Student ID	Lesson Bundle ID	...																				

13) Admin Login

Login Form

Logo	Login
Textboxes These allow the user to enter information.	User name: <input type="text"/> Password: <input type="password"/>
<input type="button" value="Return"/>	<input type="button" value="Login"/>

14) Add Field

Add new field

Labels + textboxes
Labels will update to act as headers for the information associated with the selected table.

Placeholder: ID
 Title
 Description

Buttons
will run the validation check for the entered information.

Calendar
Provides users with the ability to select a date value.

Combo boxes
These provide the user with the ability to select a value from a collection of possible answers.

Check Box
This will provide users with the ability to add True or False information.

Data Grid
The DataGrid will display information regarding records from another table.

15) Delete Confirmation

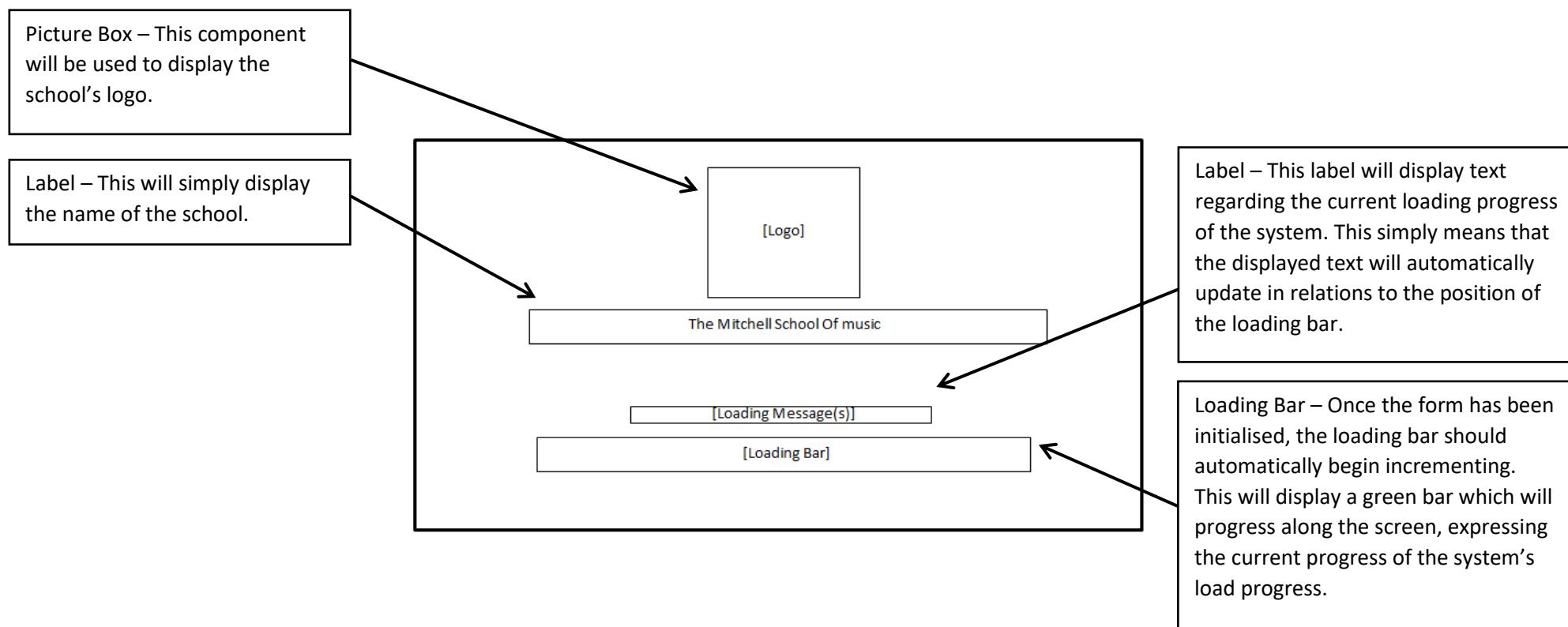
Confirmation Form

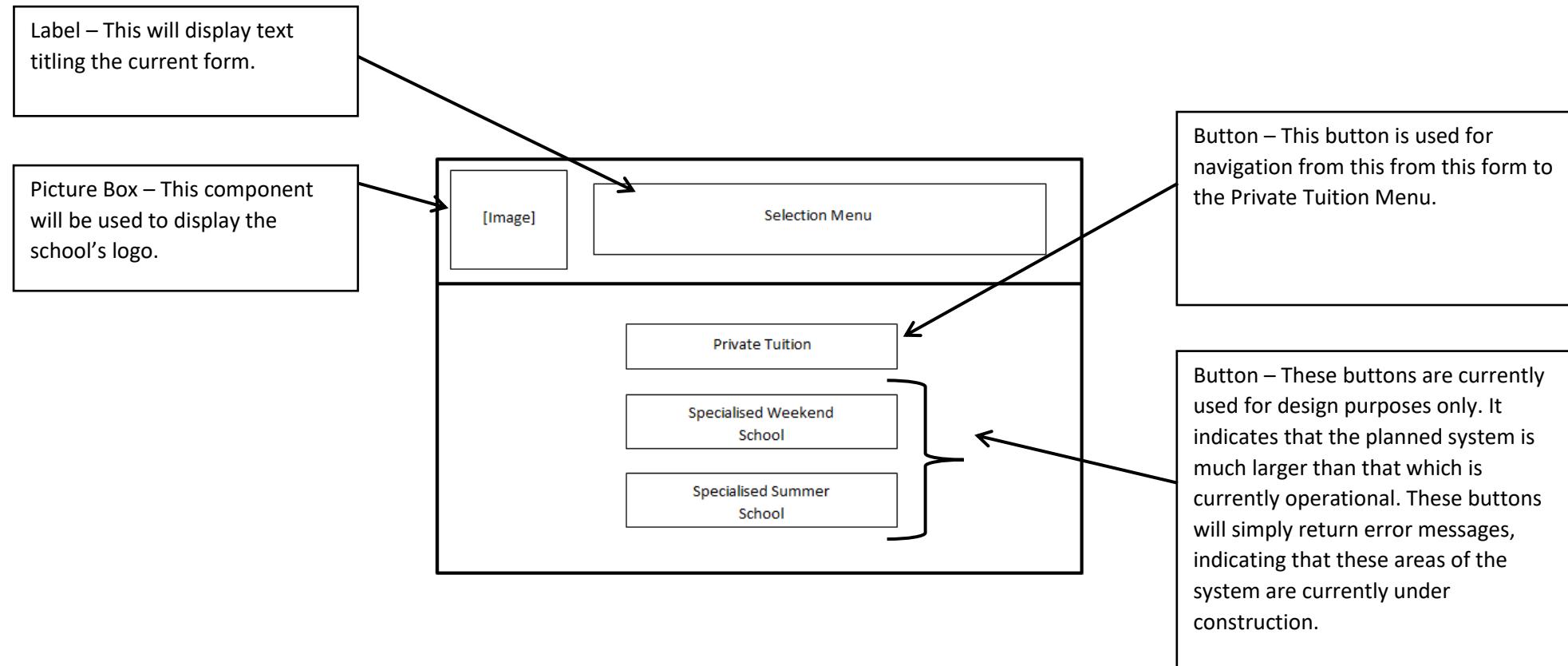
Text Boxes
these will store information regarding the selected record.

Buttons
cancel the deletion process, removing the selected record.

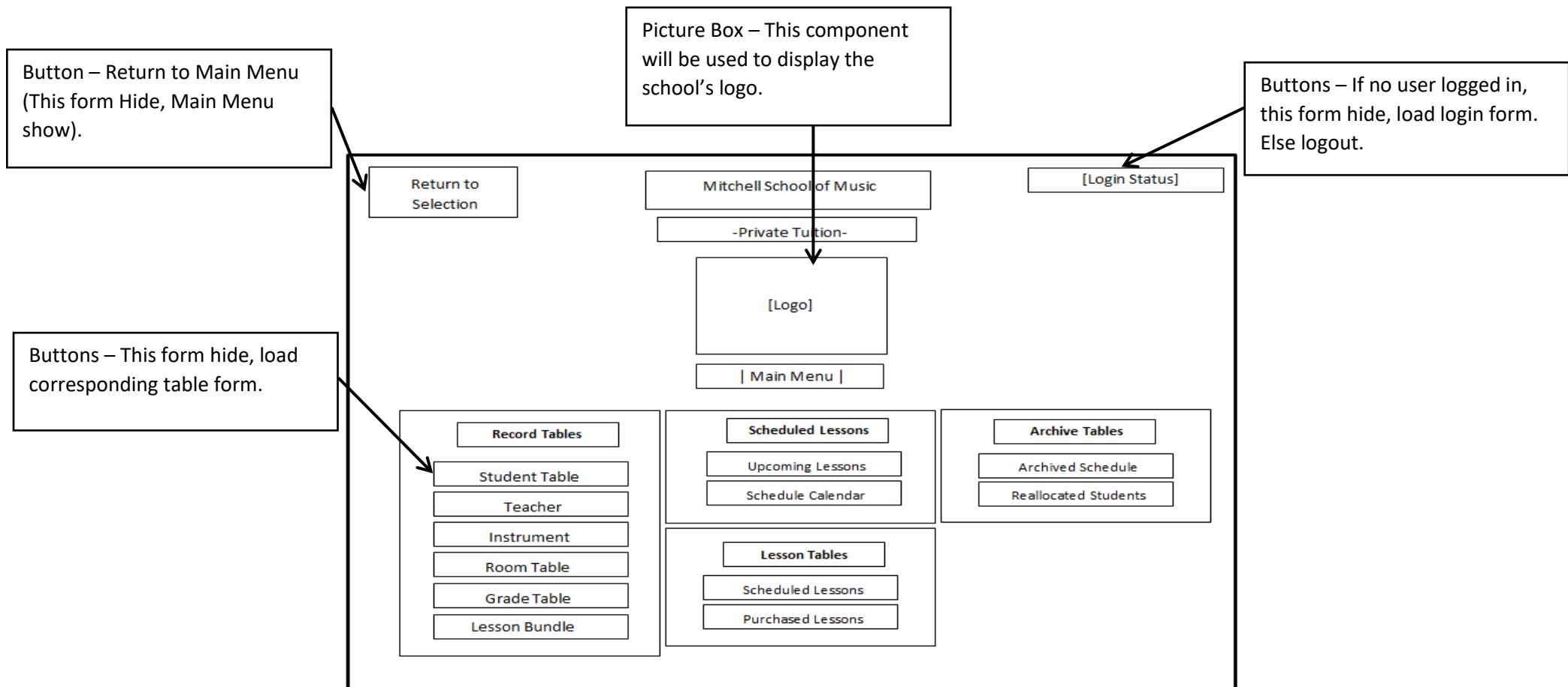
Final Story Boards

1) Splash Screen

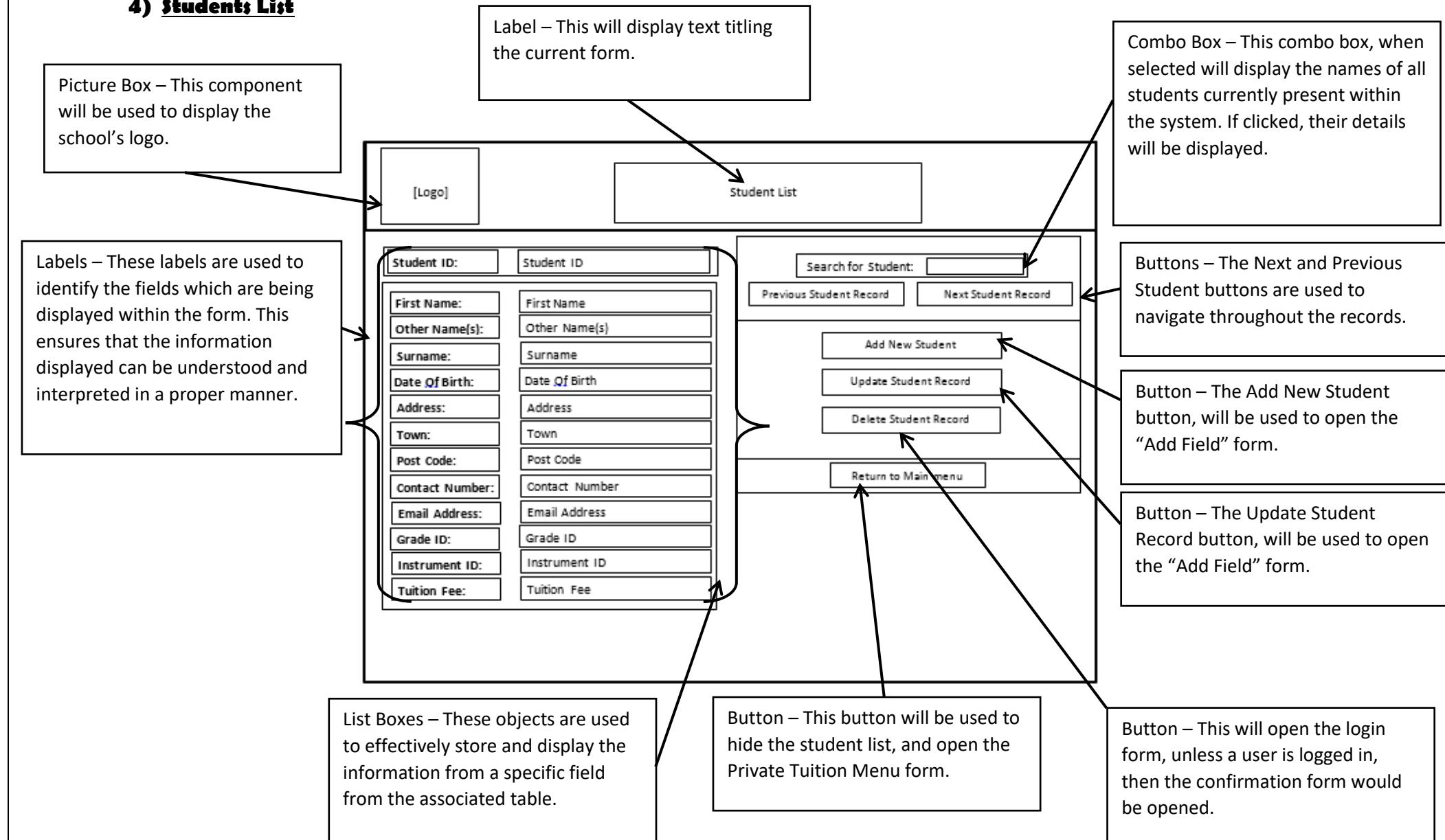


2) Selection Menu

3) Private Tuition Menu



4) Students List



5) Teachers List

Picture Box – This component will be used to display the school's logo.

Label – This will display text titling the current form.

Combo Box – This combo box, when selected will display the names of all Teachers currently present within the system. If clicked, their details will be displayed.

Labels – These labels are used to identify the fields which are being displayed within the form. This ensures that the information displayed can be understood and interpreted in a proper manner.

Teacher ID:	Teacher ID
First Name:	First Name
Surname:	Surname
Address:	Address
Town:	Town
Post Code:	Post Code
Contact number:	Contact Number
Email Address:	Email Address
Specialisation:	Specialisation
Room ID:	Room ID

Search for Student:	
Previous Teacher Record	Next Teacher Record
Add New Teacher	
Update Teacher Record	
Delete Teacher Record	
Return to Main menu	

List Boxes – These objects are used to effectively store and display the information from a specific field from the associated table.

Button – This button will be used to hide the Teacher list, and open the Private Tuition Menu form.

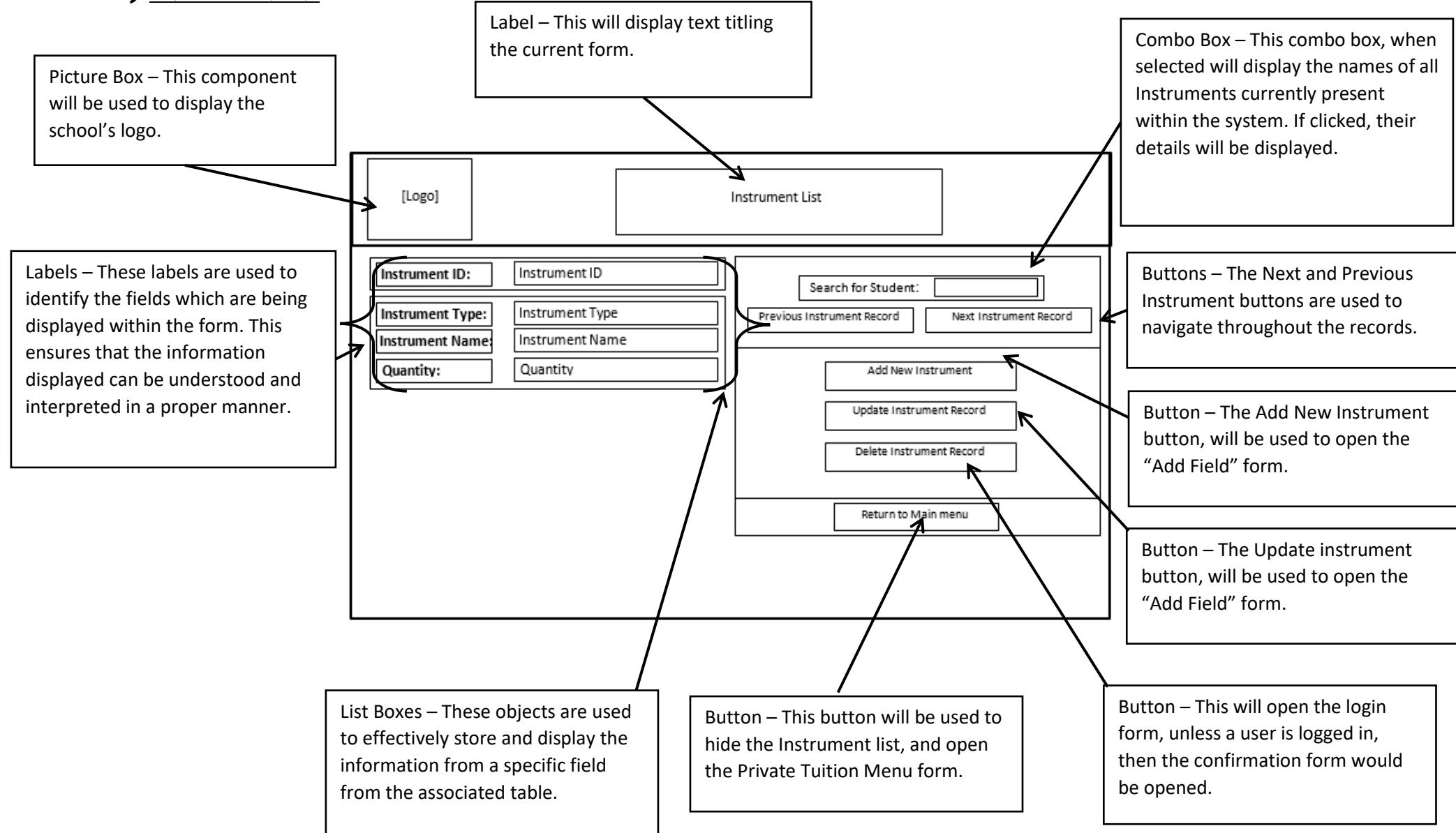
Buttons – The Next and Previous Teacher buttons are used to navigate throughout the records.

Button – The Add New Teacher button, will be used to open the “Add Field” form.

Button – The Update Teacher Record button, will be used to open the “Add Field” form.

Button – This will open the login form, unless a user is logged in, then the confirmation form would be opened.

6) Instruments List



7) Grade List

Picture Box – This component will be used to display the school's logo.

Labels – These labels are used to identify the fields which are being displayed within the form. This ensures that the information displayed can be understood and interpreted in a proper manner.

List Boxes – These objects are used to effectively store and display the information from a specific field from the associated table.

Label – This will display text titling the current form.

Combo Box – This combo box, when selected will display the names of all Grade currently present within the system. If clicked, their details will be displayed.

The Grade List form interface consists of the following components:

- Logo:** A placeholder for the school's logo.
- Title:** "Grade List".
- Grade Record List:** A table-like structure showing three rows of grade data, each with two columns. The first column contains labels: "Grade ID:", "Grade Level:", and "Grade Fee:". The second column contains corresponding input fields: "Grade ID", "Grade Level", and "Grade Fee".
- Search and Navigation:** Includes a "Search for Student:" input field and two buttons: "Previous Grade Record" and "Next Grade Record".
- Action Buttons:** A vertical stack of buttons for managing grade records: "Add New Grade", "Update Grade Record", and "Delete Grade Record".
- Return Button:** "Return to Main menu".

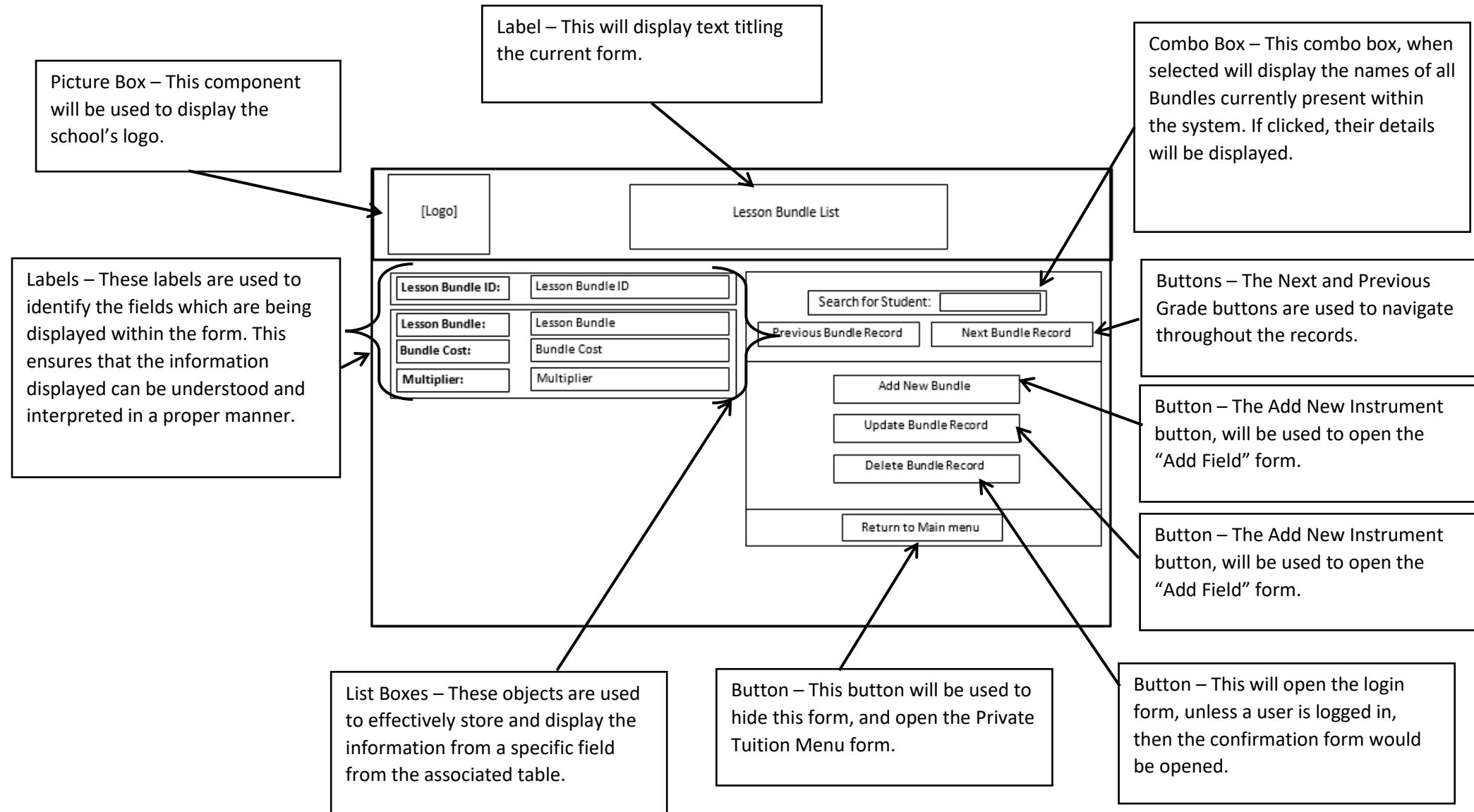
Buttons – The Next and Previous Grade buttons are used to navigate throughout the records.

Button – The Add New Grade button, will be used to open the "Add Field" form.

Button – The Update Grade button, will be used to open the "Add Field" form.

Button – This will open the login form, unless a user is logged in, then the confirmation form would be opened.

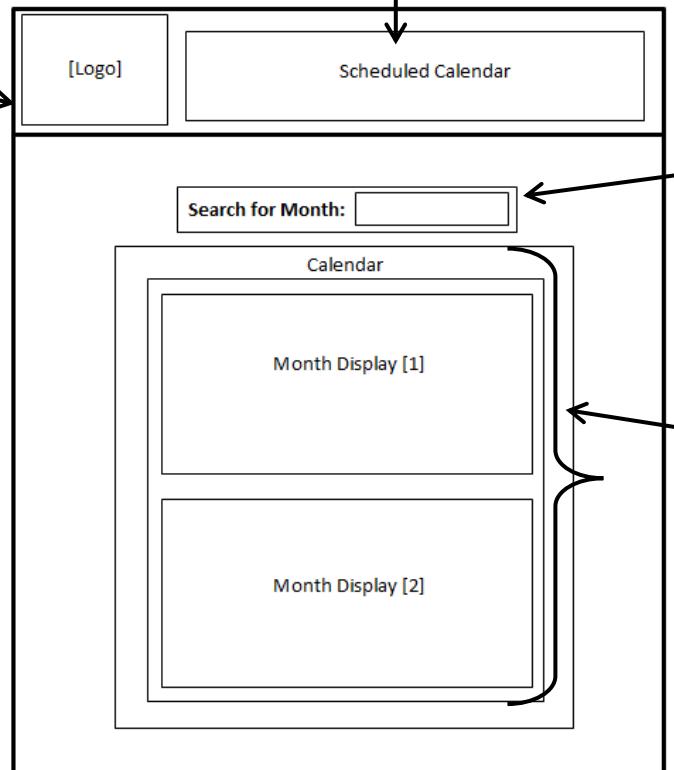
8) Lesson Bundle List



9) Schedule Calendar [Date Selection]

Picture Box – This component will be used to display the school's logo.

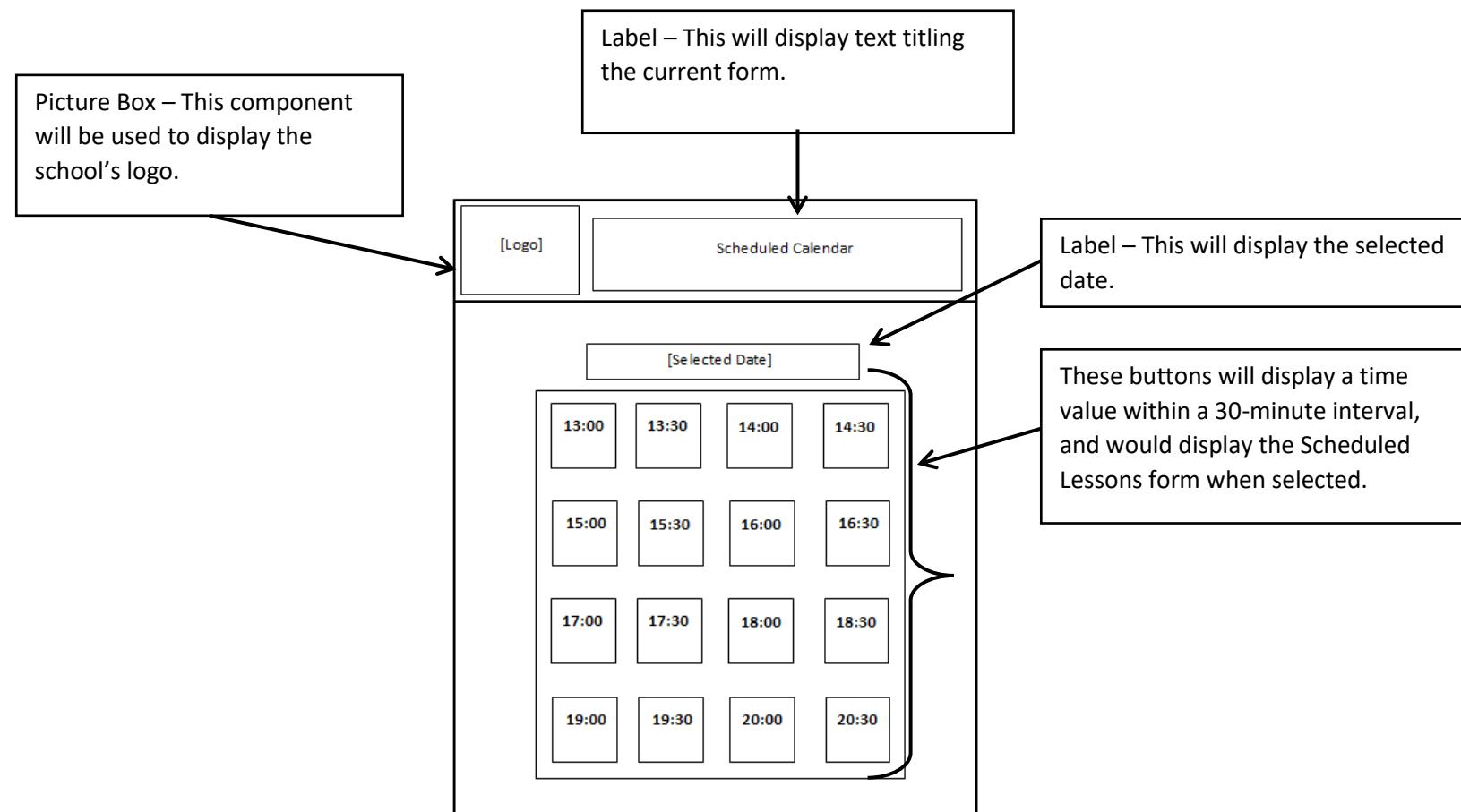
Label – This will display text titling the current form.



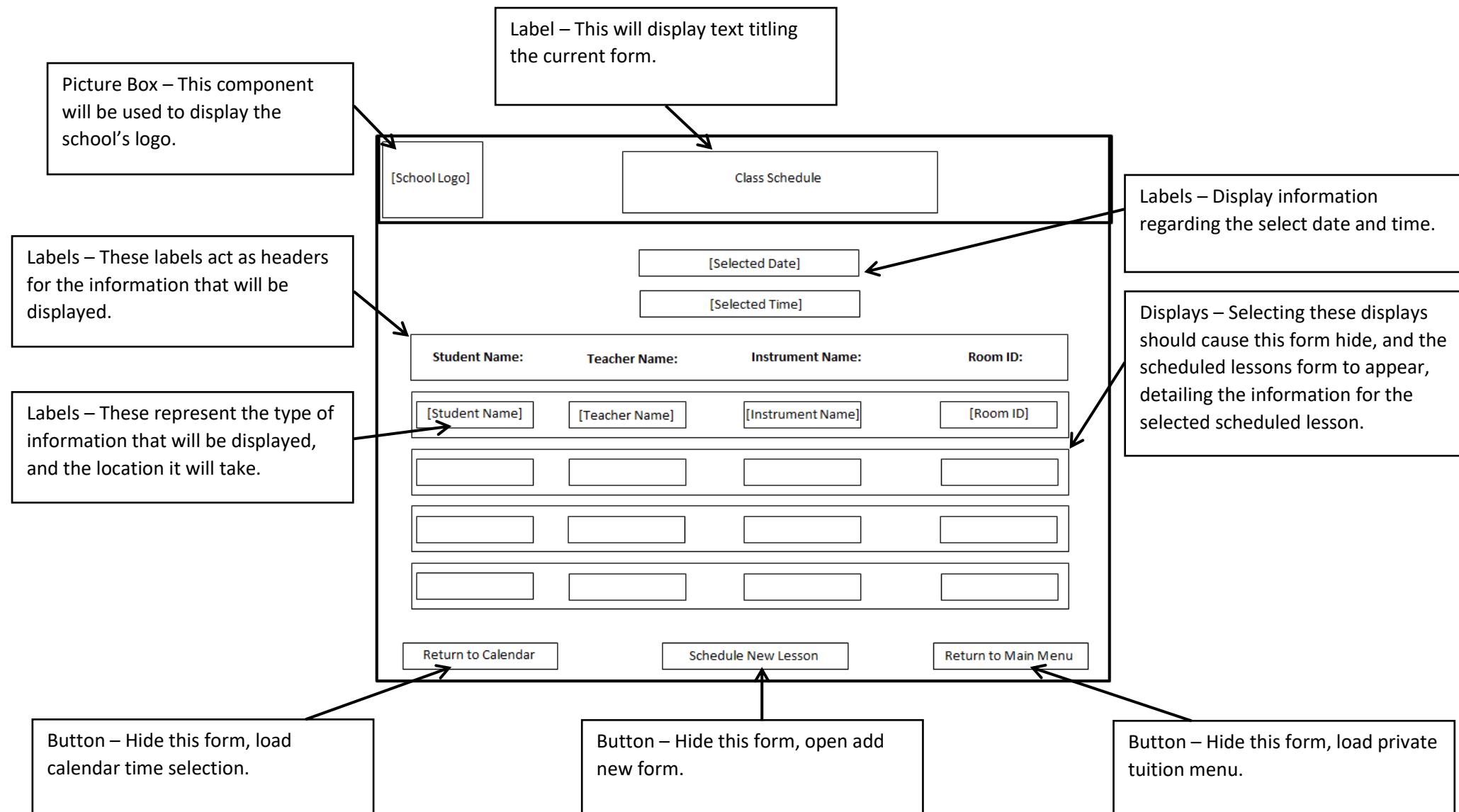
Combo box- This component will provide a drop-down menu, detailing all of the months within the year. If selected, the calendar will automatically locate the month within the current year.

Calendar - This component simply displays all of the calendar information, including months, dates, days etc. In this instance, the display has been segmented into two months, one on top, and another below.

When the user selects one of the dates present within this calendar, the Time selection form will be displayed.

10) Schedule Calendar [Time Selection]

11) Schedule Calendar [Scheduled Lessons]



12) Schedule Lessons List

[School Logo] Scheduled Lessons List

Student ID:	Student ID
First Name:	First Name
Surname:	Surname
Contact Number:	Contact Number
Grade ID:	Grade ID
Grade:	Grade

Schedule ID:	[Schedule ID]						
Schedule ID:	Student ID:	Teacher ID:	Lesson Bundle ID:	No. of Weeks	Start Date:	Booked Day(s):	Booked Time:

Search for Student:
 Previous Student Record Next Student Record
 Add New Lesson
 Update Lesson Record
 Delete Lesson Record
 Return to Main menu

Picture Box – This component will be used to display the school's logo.

Label – This will display text titling the current form.

Labels – These labels are used to identify the fields which are being displayed within the form. This ensures that the information displayed can be understood and interpreted in a proper manner.

List Boxes – These objects are used to effectively store and display the information from a specific field from the associated table.

Data Grid – Displays information regarding scheduled lesson records, in association to the selected student record.

Combo Box – This combo box, when selected will display the names of all students currently present within the system. If clicked, their details will be displayed.

Buttons – The Next and Previous Grade buttons are used to navigate throughout the records.

Button – The Add New Instrument button, will be used to open the "Add Field" form.

Button – The Add New Instrument button, will be used to open the "Add Field" form.

Button – This button will be used to hide this form, and open the Private Tuition Menu form.

Button – This will open the login form, unless a user is logged in, then the confirmation form would be opened.

13) Purchased Lessons List

[School Logo]

Scheduled Lessons List

Student ID:	Student ID	Bundle ID:	Bundle ID	Search for Student:
First Name:	First Name	Lesson Bundle:	Lesson Bundle	Previous Student Record
Surname:	Surname	Bundle Cost:	Bundle Cost	Next Student Record
Contact Number:	Contact Number	Multiplier:	Multiplier	
Grade ID:	Grade ID			Add New Lesson
Grade:	Grade			Update Lesson Record
Schedule ID:	[Schedule ID]			Delete Lesson Record
Lesson ID:	Student ID:	Bundle ID:	Payment Date:	Payment Method:
				Payment Received:
				Received Date:
				Bundle Cost

Data Grid – Displays information regarding scheduled lesson records, in association to the selected student record.

Button – This button will be used to hide this form, and open the Private Tuition Menu form.

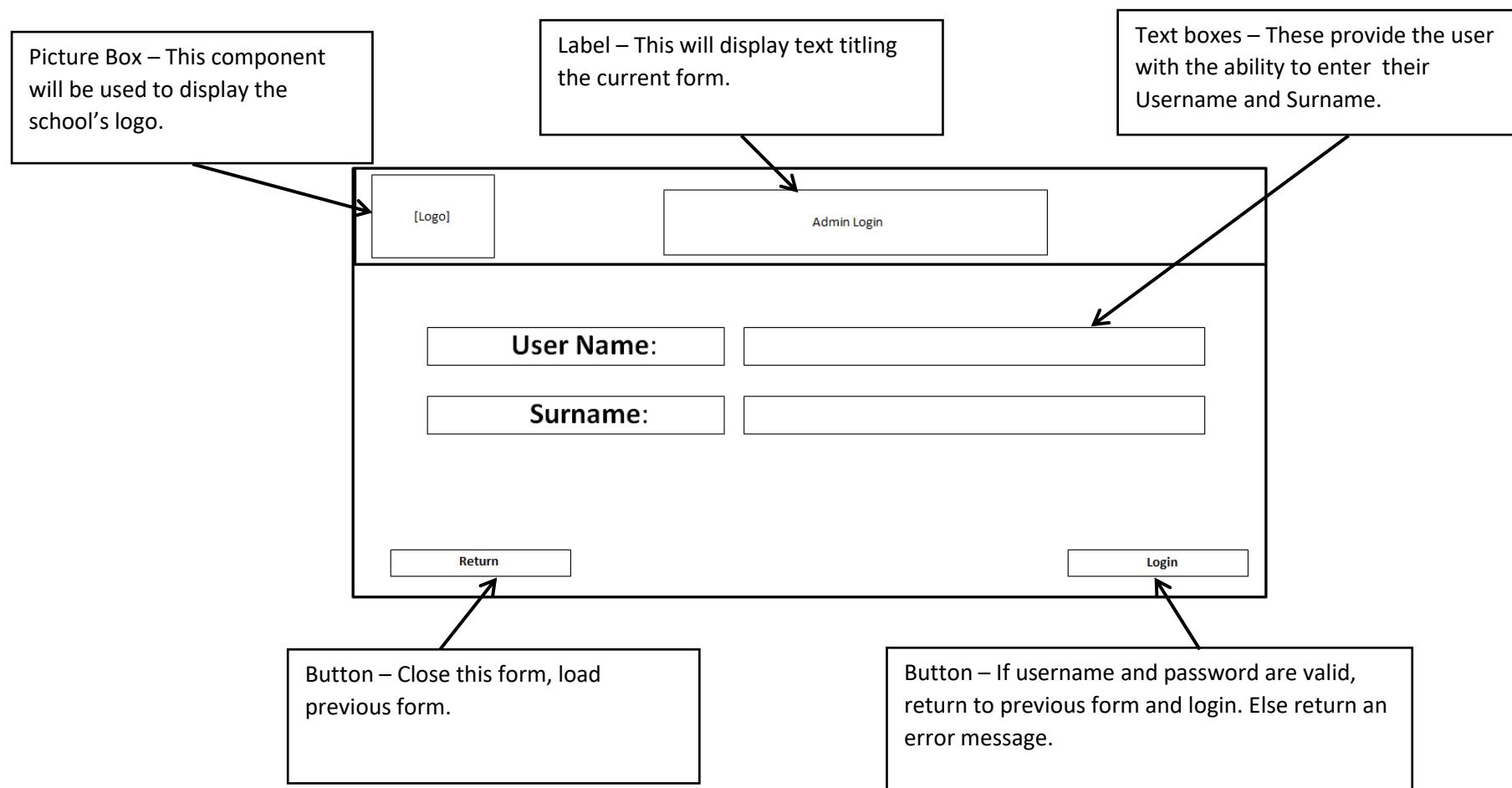
Combo Box – This combo box, when selected will display the names of all Students currently present within the system. If clicked, their details will be displayed.

Buttons – The Next and Previous Grade buttons are used to navigate throughout the records.

Button – The Add New Instrument button, will be used to open the “Add Field” form.

Button – The Add New Instrument button, will be used to open the “Add Field” form.

Button – This will open the login form, unless a user is logged in, then the confirmation form would be opened.

14) Admin login

15) Add Field

Picture Box – This component will be used to display the school's logo.

Label – This will display text titling the current form.

Button – Selecting this button will cause the previous form used to access this form to load.

Labels – These will update in corrispondance to the selected table, and will be used to title the inforamtion.

Text Boxes – These are the standarad components used to collect inforamtion from the user.

Labels – These error messages will update to display validation errors found within the entered inforamtion.

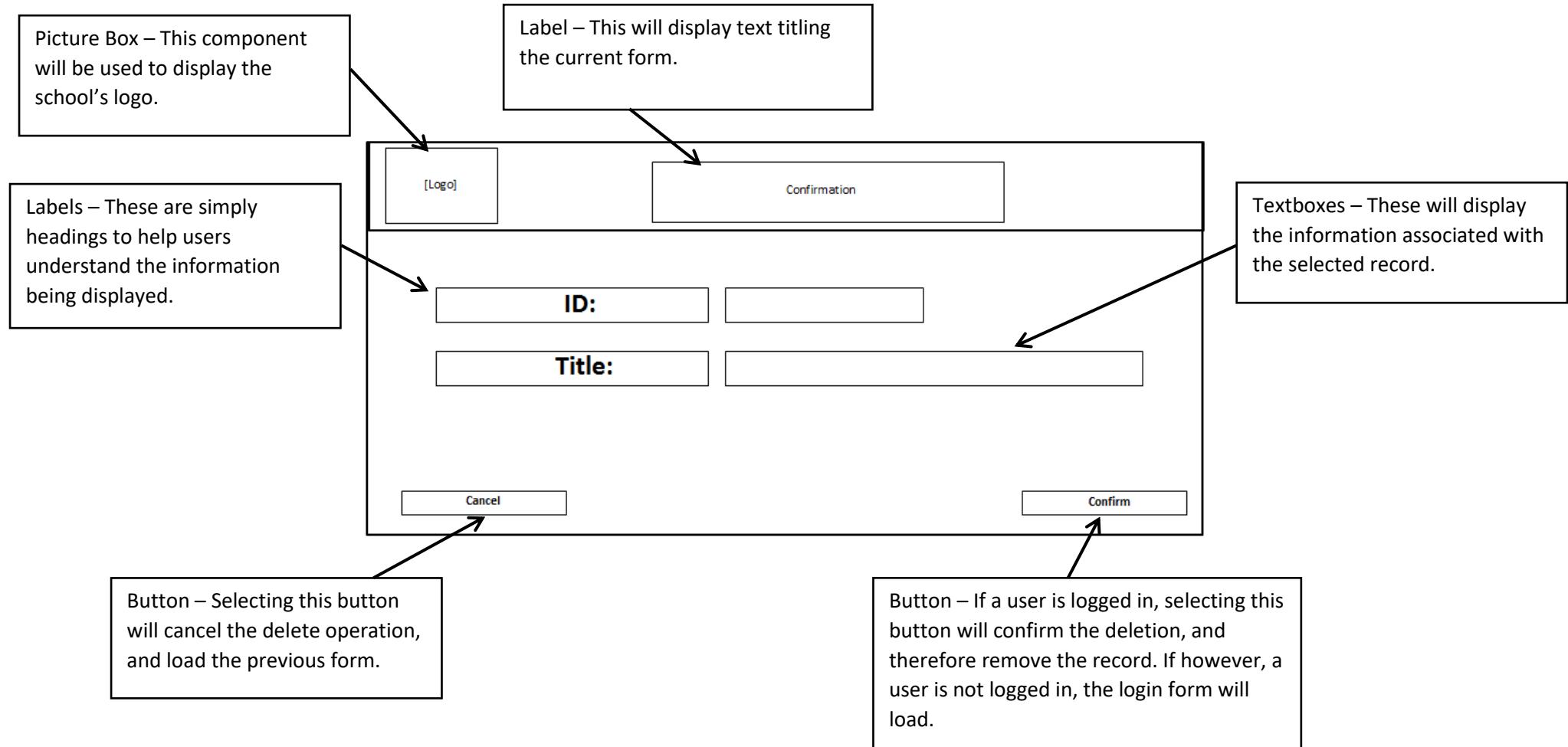
Combo Boxes – Under certain circumstnaces, these components will take the place of the displayed textboxes, and would contain associated inforamtinon.

Buttons – These buttons will be found next to specific fields to access associated menus including (Calendar, Day selection, and Dat Grid)

Check Box – This is a simppe component which will be used to represent fields which use the BIT data type.

Calendar – This component will display in association to date selection fields, and provides the user with the ability to easily select a valid value.

Data Grid – This component will be used to display inforamtion present present within an associated table.

16) Delete Confirmation

Pseudo code

Splash Screen

OUTPUT Loading Bar Progression

Loading Bar progression increments

```
IF (Progress < 25)
    {Loading Message = "Initialising program..."}
Else IF (Progress > 25 and Progress < 75)

    {Loading Message = "Organising Files..."}
Else IF (Loading Bar = 100)
    {Open Main Menu form}
```

Main Menu

INPUT "User selects Private Tuition button"

= This form will close, and Private Tuition Menu will appear.

INPUT "user Selects Specialist weekend or summer school"

= Message box appears, stating that these areas are under construction.

Private Tuition Menu

INPUT "User selects Return Button"

= This form will close, and the Main Menu will appear.

INPUT "User selects Student Table button"

= This form will close, and the Student Form will appear.

INPUT "User selects Teacher Table button"

= This form will close, and Teacher Form will appear.

INPUT "User selects Instrument Table button"

= This form will close, and Instrument Form will appear.

INPUT "User selects Grade Table button"

= This form will close, and Grade Form will appear.

INPUT "User selects Lesson Bundle Table button"

= This form will close, and Lesson Bundle Form will appear.

INPUT "User selects Room Table button"

= This form will close, and Room Form will appear.

INPUT "User selects Purchased Lessons Table button"

= This form will close, and Purchased Lessons form will appear.

INPUT "User selects Scheduled Lessons Table button"
 = This form will close, and Scheduled Lessons form will appear.

INPUT "User selects Schedule Calendar button"
 = This form will close, and Schedule Calendar (Dates) will appear.

OUTPUT Login Status
IF (user Logged in)
 {"... Logged In" (using Label) AND "Logout" [For Login Button Text]}
Else
 {"No user logged in" (using Label) AND "Login" [For Login button Text]}

INPUT "User selects Login button"
IF (User not logged in)
 {This form closes, and the Login form appears}
Else
 {UPDATE Login Status AND STORED value to False}

Student Form

ACCESS Student Table

-(From School Database)

STORE Student Information in individual lists

-(First name list, surname list etc.)

OUTPUT Student Details

-(Display information in List boxes)

OUTPUT Student Names

-(In combo box / Search box)

INPUT "user clicks Next"
 = Display next student details,

INPUT "User clicks Previous"
 = Display Previous student details,

INPUT "User Clicks Add New button"
 = Display Add New Form, close Student form,

INPUT "User Clicks Delete button"
IF (User is logged in)
THEN { Open Confirmation form }
ELSE { Open Login Form }
INPUT "User selects student from search box"
 = Display selected student details,
INPUT "User selects 'Return to Main Menu' button"
 = Close Student form, and open Private Tuition Menu.

Teacher Form

ACCESS Teacher Table
 (From School Database)

STORE Teacher Information in individual lists
 -(First name list, surname list etc.)

OUTPUT Teacher Details
 -(Display information in List boxes)

OUTPUT Teacher Names
 -(In combo box / Search box)

INPUT "user clicks Next"
 = Display next Teacher details,

INPUT "User clicks Previous"
 = Display Previous Teacher details,

INPUT "User Clicks Add New button"
 = Display Add New Form, close Teacher form,

INPUT "User Clicks Delete button"
IF (User is logged in)
THEN { Open Confirmation form }
ELSE { Open Login Form }

INPUT "User selects Teacher from search box"
 = Display selected Teachers details,

INPUT "User selects 'Return to Main Menu' button"
 = Close Teacher form, and open Private Tuition Menu.

Instrument Form

ACCESS Instrument Table
 (From School Database)

STORE Instrument Information in individual lists
 -(Instrument Type list, Instrument Name list etc.)

OUTPUT Instrument Details
 -(Display information in List boxes)

OUTPUT Instrument Names
 -(In combo box / Search box)

INPUT "user clicks Next"
 = Display next Instrument details,

INPUT "User clicks Previous"
 = Display Previous Instrument details,

INPUT "User Clicks Add New button"
 = Display Add New Form, close Instrument form,

INPUT "User Clicks Delete button"
IF (User is logged in)

THEN { Open Confirmation form }
ELSE { Open Login Form }
INPUT "User selects Instrument from search box"
 = Display selected student details,
INPUT "User selects 'Return to Main Menu' button"
 = Close Instrument form, and open Private Tuition Menu.

Grade Form

ACCESS Grade Table
 (From School Database)
STORE Grade Information in individual lists
 -(Grade list, Grade Fee List etc.)
OUTPUT Grade Details
 -(Display information in List boxes)
OUTPUT Grades
 -(In combo box / Search box)

INPUT "user clicks Next"
 = Display next Grade details,
INPUT "User clicks Previous"
 = Display Previous Grade details,
INPUT "User Clicks Add New button"
 = Display Add New Form, close Grade form,
INPUT "User Clicks Delete button"
IF (User is logged in)
THEN { Open Confirmation form }
ELSE { Open Login Form }
INPUT "User selects Grade from search box"
 = Display selected Grades details,
INPUT "User selects 'Return to Main Menu' button"
 = Close Grade form, and open Private Tuition Menu.

Lesson Bundle Form

ACCESS Lesson Bundle Table
 (From School Database)
STORE Lesson Bundle Information in individual lists
 -(Bundle List, Bundle Fee List etc.)
OUTPUT Lesson Bundle Details
 -(Display information in List boxes)
OUTPUT Lesson Bundle Names
 -(In combo box / Search box)

INPUT "user clicks Next"

= Display next Lesson Bundle details,

INPUT "User clicks Previous"

= Display Previous Lesson Bundle details,

INPUT "User Clicks Add New button"

= Display Add New Form, close Lesson Bundle form,

INPUT "User Clicks Delete button"

IF (User is logged in)

THEN { Open Confirmation form }

ELSE { Open Login Form }

INPUT "User selects Lesson Bundle from search box"

= Display selected Lesson Bundle details,

INPUT "User selects 'Return to Main Menu' button"

= Close Lesson Bundle form, and open Private Tuition Menu.

Room Form

ACCESS Room Table

(From School Database)

STORE Room Information in individual lists

-(Room type List, Specialisation list etc.)

OUTPUT Room Details

-(Display information in List boxes)

OUTPUT Room Number

-(In combo box / Search box)

INPUT "user clicks Next"

= Display next Room details,

INPUT "User clicks Previous"

= Display Previous Room details,

INPUT "User Clicks Add New button"

= Display Add New Form, close Room form,

INPUT "User Clicks Delete button"

IF (User is logged in)

THEN { Open Confirmation form }

ELSE { Open Login Form }

INPUT "User selects Room from search box"

= Display selected Rooms details,

INPUT "User selects 'Return to Main Menu' button"

= Close Room form, and open Private Tuition Menu.

Purchased Lessons Form

ACCESS Purchased Lessons Table
 (From School Database)

ACCESS Student Table
 (From School Database)

ACCESS Grade Table
 (From School Database)

ACCESS Lesson Bundle Table
 (From School Database)

STORE Purchased Lessons Information in individual lists:

- (Purchased Lesson Student ID List, Purchased Date, etc.)

STORE Student Information in individual lists:

- (Student First Name List, Student Surname List etc.)

STORE Grade Information in individual lists:

- (Grade List, Grade Fee List etc.)

STORE Lesson Bundle Information in individual lists:

- (Lesson Bundle list, Lesson Bundle Fee list etc.)

OUTPUT Purchased Lessons Information:

- (Display information in Data Grid)

OUTPUT Student Information:

- (In combo box / Search box)

OUTPUT Grade information:

- (In combo box / Search box)

OUTPUT Lesson Bundle Information:

- (In combo box / Search box)

INPUT "user clicks Next"

- = Display Grade, Lesson Bundle and Student information for next Student, with associated Purchased lesson information.

INPUT "User clicks Previous"

- = Display Grade, Lesson Bundle and Student information for previous Student, with associated Purchased lesson information.

INPUT "User Clicks Add New button"

- = Display Add New Form, close Purchased Lessons form,

INPUT "User Clicks Delete button"

- IF** (User is logged in)

- THEN** { Open Confirmation form }

- ELSE** { Open Login Form }

INPUT "User selects Student from search box"

- = Display Grade, Lesson Bundle and Student information for Selected Student, with associated Purchased lesson information.

INPUT "User selects 'Return to Main Menu' button"

- = Close Purchased Lessons form, and open Private Tuition Menu.

Scheduled Lessons Form

ACCESS Scheduled Lessons Table
 (From School Database)

ACCESS Student Table
 (From School Database)

ACCESS Teacher Table
 (From School Database)

ACCESS Grade Table
 (From School Database)

STORE Scheduled Lessons Information in individual lists:

- (Purchased Lesson Student ID List, Purchased Date, etc.)

STORE Student Information in individual lists:

- (Student First Name List, Student Surname List etc.)

STORE Grade Information in individual lists:

- (Grade List, Grade Fee List etc.)

STORE Teacher Information in individual lists:

- (Teacher First Name list, Teacher Surname list etc.)

OUTPUT Scheduled Lessons Information:

- (Display information in Data Grid)

OUTPUT Student Information:

- (In combo box / Search box)

OUTPUT Grade information:

- (In combo box / Search box)

OUTPUT Teacher Information:

- (In combo box / Search box)

INPUT "user clicks Next"

- = Display Grade, Teacher and Student information for next Student, with associated Purchased lesson information.

INPUT "User clicks Previous"

- = Display Grade, Teacher and Student information for previous Student, with associated Purchased lesson information.

INPUT "User Clicks Add New button"

- = Display Add New Form, close Scheduled Lessons form,

INPUT "User Clicks Delete button"

- IF** (User is logged in)

- THEN** { Open Confirmation form }

- ELSE** { Open Login Form }

INPUT "User selects Student from search box"

- = Display Grade, Teacher and Student information for Selected Student, with associated Purchased lesson information.

INPUT "User selects 'Return to Main Menu' button"

- = Close Purchased Lessons form, and open Private Tuition Menu.

INPUT "User selects 'Update End Dates' button"
 = Calculate end date value, then update display

Schedule Calendar (Dates)

OUTPUT Calendar Display
 (Individual days and months)
OUTPUT All Month Names
 [In combo box / Search box]

INPUT "User selects month from search box"
 = Update calendar to display selected month
INPUT "User selects day"
 = **STORE** Selected date, and then proceed to open "Schedule Calendar (Times)", and close "Schedule Calendar (Dates)."
INPUT "User selects 'Return to Main Menu' button"
 = Close Schedule Calendar (Dates), and open Private Tuition Menu

Schedule Calendar (Times)

OUTPUT Time Display
 (4X4 grid of buttons, each representing a time interval of 30 minutes)
OUTPUT Selected Date
 [Present in label]

INPUT "User selects Time based button"
 = **STORE** Selected Time value, then proceed to open "Schedule Calendar (Lessons)", and close "Schedule Calendar (Times)".
INPUT "user selects 'Return to Main Menu' button"
 = Close, Schedule Calendar (Times), and open Private Tuition Menu.

Schedule Calendar (Lesson Details)

STORE any lesson records that match the selected date and time.

OUTPUT selected Day and Time
 [Using Labels]
IF (There is at least one scheduled lesson found)
 {**OUTPUT** Scheduled lesson information}
 [use labels to display: Student Name, Teacher Name, Instrument Name etc.]
ELSE

{Display Error message}
 [Message is "No scheduled classes"]

- INPUT** "User selects Schedule New Lesson"
 = **STORE** Previous Field value as "Scheduled lessons", then proceed to open the Add field form, and close Schedule Calendar (Lesson Details)
- INPUT** "User selects Return to Calendar"
 = **CLEAR** stored time information, close this form, and then open Calendar (Times)
- INPUT** "User selects Return to Main Menu"
 = **CLEAR** stored information, close this form, and then open Private Tuition Menu.
- INPUT** "User selects a displayed lesson record"
 = **UNIQUELY STORE** Details corresponding to selected record, close this form, then proceed to open scheduled lessons form, and display the selected record.

Deletion Confirmation

STORE Previous Form
 STORE Selected Record Information
 OUTPUT Selected Record Information
INPUT "User Selects Confirm"
 {Delete Selected Record}
INPUT "User Selects Return"
 {Return to previous form}

Login

INPUT "User Selects Login"
IF (Username and Password are valid)
 {Login}
ELSE
 {Return Error Message}

Add New Field Form

IF (Previous Form value = "Students")
 { **OUTPUT** Corresponding contents}
 [Textboxes, Calendars, combo boxes etc.]
{OUTPUT Names and titles}
 [Title= "New Student", Add button = "Add New Student", Return = "Return to Students form" etc.]

IF (Previous Form value = "Teachers")

{ **OUTPUT** Corresponding contents}

[Textboxes, Calendars, combo boxes etc.]

{**OUTPUT** Names and titles}

[Title= "New Teacher", Add button = "Add New Teacher", Return = "Return to Teachers form" etc.]

IF (Previous Form value = "Instruments")

{ **OUTPUT** Corresponding contents}

[Textboxes, Calendars, combo boxes etc.]

{**OUTPUT** Names and titles}

[Title= "New Instrument", Add button = "Add New Instrument", Return = "Return to Instruments form" etc.]

IF (Previous Form value = "Room")

{ **OUTPUT** Corresponding contents}

[Textboxes, Calendars, combo boxes etc.]

{**OUTPUT** Names and titles}

[Title= "New Room", Add button = "Add New Room", Return = "Return to Rooms form" etc.]

IF (Previous Form value = "Grade")

{ **OUTPUT** Corresponding contents}

[Textboxes, Calendars, combo boxes etc.]

{**OUTPUT** Names and titles}

[Title= "New Grade", Add button = "Add New Grade", Return = "Return to Grades form" etc.]

IF (Previous Form value = "Lesson Bundle")

{ **OUTPUT** Corresponding contents}

[Textboxes, Calendars, combo boxes etc.]

{**OUTPUT** Names and titles}

[Title= "New Lesson Bundle", Add button = "Add New Lesson Bundle", Return = "Return to Lesson Bundle form" etc.]

IF (Previous Form value = "Purchased Lessons")

{ **OUTPUT** Corresponding contents}

[Textboxes, Calendars, combo boxes etc.]

{**OUTPUT** Names and titles}

[Title= "New Purchased Lessons", Add button = "Add New Purchased Lessons", Return = "Return to Purchased Lessons form" etc.]

IF (Previous Form value = "Scheduled Lessons")

{ **OUTPUT** Corresponding contents}

[Textboxes, Calendars, combo boxes etc.]

{**OUTPUT** Names and titles}

[Title= "New Scheduled Lessons", Add button = "Add New Scheduled Lessons",

Return = "Return to Scheduled Lessons form" etc.]

IF (Purpose = "UPDATE")

{**OUTPUT** Selected Record Details}

OUTPUT Update Form Displays}

INPUT "User Selects Add New Field"

{Run Associated Validation}

IF (All Information Valid)

{Add New Record to associated Table}

INPUT "User Selects Update"

{Run Associated Validation}

IF (All Information Valid)

{Update Record}

| Chapter 4 |

| Testing |

Test Plan

2) Forms**2.1) Splash Screen**

frmSplash							
Test Number	Tested Data	Test Purpose	Additional Details	Expected Outcome	Actual Outcome	Corrective action	Page of Evidence
Process [A]:		Loading Bar Progress + Feedback					
2.1[A][1]	Loading Bar value	This test would be used to ensure that the Loading Bar value increments as intended.	Each Timer tick: LoadingBar.Increment(2); UpdateLoadMessage();	The bar should visually express the increment.	As Expected.	N/A	
2.1[A][2]	Load Message Update	To ensure that the error message updates accordingly to the progress of the loading bar.	If LoadingValue > 0 AND < 25: Message = "Initialising Program..." Else if Loading Value > 25 AND < 75: Message = "Organising Files" Else	As the bar increments, the loading messages should update in accordance to its progress.	As Expected.	N/A	

			Message = "Finalising Program..." OpenMenu(); lblLoadingProgress.Text = LoadMessage;				
2.1[A][3]	Load Main Form	To ensure that upon reaching 100%, the Main Menu form opens.	If Loading value = 100, OpenMenu();	Upon reaching 100, the splash form should hide, and the Main form should appear.	As Expected.	N/A	
Process: [B]:		Database Information Access + Storage					
2.1[B][1]	Student Storage Lists	To ensure that all information is stored correctly	SQL Command = "SELECT * FROM Students";	All student information should be stored in individual lists. The values present within these lists should then share a common index, making it	As Expected.	N/A	

				possible to retrieve entire records.			
2.1[B][2]	Lesson Dates Storage Lists	To ensure that all information is stored correctly	SQL Command = "SELECT * FROM LessonDates";	All Lesson Date information should be stored in individual lists. The values present within these lists should then share a common index, making it possible to retrieve entire records.	As Expected.	N/A	
2.1[B][3]	Lesson Dates_Archive Storage Lists	To ensure that all information is stored correctly	SQL Command = "SELECT * FROM LessonDates_Archive";	All Archived Lesson Date information should be stored in individual lists. The values present within	As Expected.	N/A	

				these lists should then share a common index, making it possible to retrieve entire records.			
2.1[B][4]	Scheduled Lessons Storage Lists	To ensure that all information is stored correctly	SQL Command = "SELECT * FROM Scheduled_Lessons";	All Scheduled Lesson information should be stored in individual lists. The values present within these lists should then share a common index, making it possible to retrieve entire records.	As Expected.	N/A	
Process [C]:	LessonDate Archive						
2.1[C][1]	Archived Lesson Dates	To ensure that details are archived correctly.	When archiving, locate any records that possess an end date, less than today.	When running the program, the system should	As Expected.	N/A	

			Then append the information to the archive table, and finally remove it from the origin table.	automatically identify completed lessons, and then proceed to archive them.			
2.1[C][2]	Add New Lesson Date(s)	To test the system's response to new information.	This test will include two records. One of which dawns a pasted date, and the other a future date.	The older record should be moved to the archive table, while the newer one should be left in the origin table.	As Expected.	N/A	
Process [D]:		Clear Old LessonDate Archives:					
2.1[D][1]	Lesson Date Archive Records	To ensure that information is deleted accurately.	This method would be used to ensure that the Lesson Date Archive table does not become excessively cluttered. Records will be stored for two months, and then removed from the system completely.	The system should automatically identify any present records which possess a date which falls before (Today – 2 months), and then delete them from the system	As Expected.	N/A	

				completely.			
2.1[D][2]	Add New Archived Lesson Date(s)	To test how the system interacts with new information.	This test will include two records. One of which dawns a pasted date, and the other a future date.	The older record should be removed from the system, and the newer one should continue to resign within the Archived Lesson dates table.	As Expected.	N/A	
Process [E]:		Attendance					
2.1[E][1]	Attendance Record	To test the accuracy and general operations of the attendance record.	Student attendance is documented in correspondence to ant lessons within the previous month, from the current date. If they have marked absence from a lesson, their absences will increase by 1.	A list of student absences should appear, sharing details of that present within the lesson dates table.	As Expected.	N/A	
Process [F]:		Reallocate Students:					
2.1[F][1]	Reallocated Student	To ensure that details are	If a student possesses more than 3 absences	All students who have more than	As Expected.	N/A	

	Records archived correctly.	within a months period, they will become eligible for Reallocation.	3 absences documented, should be moved to the reallocation table.				
Process [G]:							
2.1[G][1]	Archived Scheduled Lesson Records	To ensure that details are archived correctly.	When archiving, locate any records that possess an end date, less than today. Then append the information to the archive table, and finally remove it from the origin table.	When running the program, the system should automatically identify completed lessons, and then proceed to archive them.	As Expected.	N/A	
2.1[G][2]	Add New Archived Scheduled Lesson.	To test the system's interactions with new information.	This test will include two records. One of which dawns a pasted date, and the other a future date.	The older record should be moved to the archive table, while the newer one should be left in the origin table.	As Expected.	N/A	

2.2) Main Menu

frmMain							
Test Number	Tested Data	Test Purpose	Additional Details	Expected Outcome	Actual Outcome	Corrective action	Page of Evidence
Process [A]:		Navigational Buttons					
2.2[A][1]	Private Tuition Button	To ensure that the desired operation runs when the button is selected.	frmPrivateTuition PrivateT = new frmPrivateTuition(); PrivateT.Show(); this.Close();	Upon selecting this button, the Main form should close (Hide), and the Private Tuition form should appear.	As Expected.	N/A	
2.2[A][2]	Specialist weekend school	To ensure that the desired operation runs when the button is selected.	MessageBox.Show ("System Is In Development. Please View 'Private Tuition'.")	Upon selecting this button, a message should appear, detailing that the system is under development, and that they should instead access the private tuition form.	As Expected.	N/A	
2.2[A][3]	Specialist Summer School	To ensure that the desired	MessageBox.Show ("System Is In Development.")	Upon selecting this button, a message should	As Expected.	N/A	

		operation runs when the button is selected.	Please View 'Private Tuition'.)	appear, detailing that the system is under development, and that they should instead access the private tuition form.			
--	--	---	---------------------------------	---	--	--	--

2.3) Private Tuition Menu

frmPrivateTuition							
Test Number	Tested Data	Test Purpose	Additional Details	Expected Outcome	Actual Outcome	Corrective action	Page of Evidence
Process [A]:		Navigational Buttons:					
2.3[A][1]	Students Table <Button>	To ensure that the selected button is operational.	GlobalVariables. StudentForm.Show(); this.Hide();	Selecting this button should cause this form to close, and the Students table form to appear.	As Expected.	N/A	
2.3[A][2]	Teachers Table <Button>	To ensure that the selected button is operational.	N/A	Selecting this button should cause this form to close, and the Teachers table form to appear.	As Expected.	N/A	
2.3[A][3]	Instrument Table <Button>	To ensure that the selected button is operational.	N/A	Selecting this button should cause this form to close, and the Instrument table form to appear.	As Expected.	N/A	
2.3[A][4]	Room Table <Button>	To ensure that the selected button is	N/A	Selecting this button should cause this form to close, and the Room table form	As Expected.	N/A	

		operational.		to appear.			
2.3[A][5]	Grade Table <Button>	To ensure that the selected button is operational.	N/A	Selecting this button should cause this form to close, and the Grade table form to appear.	As Expected.	N/A	
2.3[A][6]	Lesson Bundle Table <Button>	To ensure that the selected button is operational.	N/A	Selecting this button should cause this form to close, and the Lesson Bundle table form to appear.	As Expected.	N/A	
2.3[A][7]	Archived Scheduled Lessons Table <Button>	To ensure that the selected button is operational.	N/A	Selecting this button should cause this form to close, and the Archived Scheduled Lesson table form to appear.	As Expected.	N/A	
2.3[A][8]	Reallocated Students Table <Button>	To ensure that the selected button is operational.	N/A	Selecting this button should cause this form to close, and the Reallocated Students table form to appear.	As Expected.	N/A	

2.3[A][9]	Upcoming Scheduled Lessons Table <Button>	To ensure that the selected button is operational.	N/A	Selecting this button should cause this form to close, and the Upcoming Scheduled Lessons table form to appear.	As Expected.	N/A	
2.3[A][10]	Schedule Calendar Table <Button>	To ensure that the selected button is operational.	N/A	Selecting this button should cause this form to close, and the Schedule Calendar table form to appear.	As Expected.	N/A	
2.3[A][11]	Scheduled Lessons Table <Button>	To ensure that the selected button is operational.	N/A	Selecting this button should cause this form to close, and the Scheduled Lessons table form to appear.	As Expected.	N/A	
2.3[A][12]	Purchased Lessons Table <Button>	To ensure that the selected button is operational.	N/A	Selecting this button should cause this form to close, and the Purchased Lessons table form to appear.	As Expected.	N/A	

2.3[A][13]	Return to Selection Menu <Button>	To ensure that the selected button is operational.	N/A	Selecting this button should cause this form to close, and the Main form to appear.	As Expected.	N/A	
Process: [B]:							
2.3[B][1]	Login {Select} [No User]	To ensure that the selected button is operational.	This test will show the outcome of selecting the button while no user is currently logged in.	Selecting the button should cause the login form to load.	As Expected.	N/A	
2.3[B][2]	Login {Select} [User Logged In]	To ensure that the selected button is operational.	While this test, will show the outcome of selecting the button while a user is logged in.	Selecting this button, will cause the user to logout.	As Expected.	N/A	
2.1[B][3]	Login {Display} + Login Message	To ensure that the button display updates accordingly.	Depending on the login status, when clicked the button may display different information.	If the user is logged out, then the button should state "Login", otherwise it should display "Logout".	As Expected.	N/A	

2.1[B][4]	User Label	To ensure that the label updates accordingly.	N/A	Upon logging in, the label should update as to display the users name.			
------------------	------------	---	-----	--	--	--	--

2.4) Students From

frmStudentTable								
Test Number	Tested Data	Test Purpose	Additional Details	Expected Outcome	Actual Outcome	Corrective action	Page of Evidence	
Process [A]:		Information Display						
2.4[A][1]	Student Information	To ensure that all information is relevant to the selected student.	N/A	All details should correlate to the student selected by the user.	As Expected.	N/A		
Process [B]:		Record Navigation						
2.4[B][1]	Student Search	To ensure that the search function is operational.	A combo box detailing the first and last name for all students is present within the form.	Upon selecting a value present within the search combo box, the form display should update, to display the associated details.	As Expected.	N/A		
2.4[B][2]	Next Student	To ensure that the button operates as intended.	(Selected Student ID + 1)	Selecting this button should cause the form display to update, as to detail	As Expected.	N/A		

				information regarding the next student present within the table.			
2.4[B][3]	Previous Student	To ensure that the button operates as intended.	(Selected Student ID – 1)	Selecting this button should cause the form display to update, as to detail information regarding the previous student present within the table.	As Expected.	N/A	
2.4[B][4]	First Student	To ensure that the button operates as intended.	(Selected Student ID = AllStudentIDs[0])	Selecting this button should cause the form display to update, as to detail information regarding the first student record present within the table.	As Expected.	N/A	

2.4[B][5]	Last Student	To ensure that the button operates as intended.	(Selected Student ID = (AllStudentIDs. Count() – 1))	Selecting this button should cause the form display to update, as to detail information regarding the last student record present within the table.	As Expected.	N/A	
Process [C]:		Form Navigation					
2.4[C][1]	Add New Student Record	To ensure that the button operates as intended.	Global.Previous Form = “StudentForm _Add”.	Selecting this button should cause this form to close, and the Add Field Form to appear.	As Expected.	N/A	
2.4[C][2]	Update Student Record	To ensure that the button operates as intended.	Global.Previous Form = “StudentForm _Update”.	Selecting this button should cause this form to close, and the Add Field Form to appear.	All information was transferred correctly, bar the student surname.	An initial test was run upon identifying this problem, as to determine whether it was related to the storage of information, or	

2.4[C][3]	Delete Student Record [User Logged in]	To ensure that the button operates as intended.	Global.ID = (Selected Student ID) Global.FieldName = (Selected	Selecting this button should cause this form to close, and the Confirmation Form to appear.	As Expected.	N/A	

			FirstName + Surname)				
2.4[C][4]	Delete Student Record [No user logged in]	To ensure that the button operates as intended.	N/A	Selecting this button should cause this form to close, and the Login Form to appear.	As Expected.	N/A	
2.4[C][5]	Return to Main Menu	To ensure that the button operates as intended.	N/A	Selecting this button should cause this form to close, and the Private Tuition Form to appear.	As Expected.	N/A	

2.5) Teachers Form

frmTeacherTable								
Test Number	Tested Data	Test Purpose	Additional Details	Expected Outcome	Actual Outcome	Corrective action	Page of Evidence	
Process [A]:		Information Display						
2.5[A][1]	Teacher Information	To ensure that all information is relevant to the selected student.	N/A	All details should correlate to the Teacher selected by the user.	As Expected.	N/A		
Process [B]:		Record Navigation						
2.5[B][1]	Teacher Search	To ensure that the search function is operational.	A combo box detailing the first and last name for all Teachers is present within the form.	Upon selecting a value present within the search combo box, the form display should update, to display the associated details.	As Expected.	N/A		
2.5[B][2]	Next Teacher	To ensure that the button operates as intended.	(Selected Student ID + 1)	Selecting this button should cause the form display to update, as to detail	As Expected.	N/A		

				information regarding the next Teacher present within the table.			
2.5[B][3]	Previous Teacher	To ensure that the button operates as intended.	(Selected Teacher ID – 1)	Selecting this button should cause the form display to update, as to detail information regarding the previous Teacher present within the table.	As Expected.	N/A	
2.5[B][4]	First Teacher	To ensure that the button operates as intended.	(Selected Teacher ID = AllTeacherIDs[0])	Selecting this button should cause the form display to update, as to detail information regarding the first Teacher record present within the table.	As Expected.	N/A	

2.5[B][5]	Last Teacher	To ensure that the button operates as intended.	(Selected Student ID = (AllStudentIDs. Count() – 1))	Selecting this button should cause the form display to update, as to detail information regarding the last Teacher record present within the table.	As Expected.	N/A	
Process [C]:		Form Navigation					
2.5[C][1]	Add New Teacher Record	To ensure that the button operates as intended.	Global.Previous Form = “TeacherForm _Add”.	Selecting this button should cause this form to close, and the Add Field Form to appear.	As Expected.	N/A	
2.5[C][2]	Update Teacher Record	To ensure that the button operates as intended.	Global.Previous Form = “TeacherForm _Update”.	Selecting this button should cause this form to close, and the Add Field Form to appear.	As Expected.	N/A	
2.5[C][3]	Delete Teacher Record	To ensure that the button	Global.ID = (Selected Teacher	Selecting this button should	As Expected.	N/A	

	[User Logged in]	operates as intended.	ID) Global.FieldName = (Selected FirstName + Surname)	cause this form to close, and the Confirmation Form to appear.			
2.5[C][4]	Delete Teacher Record [No user logged in]	To ensure that the button operates as intended.	N/A	Selecting this button should cause this form to close, and the Login Form to appear.	As Expected.	N/A	
2.5[C][5]	Return to Main Menu	To ensure that the button operates as intended.	N/A	Selecting this button should cause this form to close, and the Private Tuition Form to appear.	As Expected.	N/A	

2.6) Instrument Form

frmSplash							
Test Number	Tested Data	Test Purpose	Additional Details	Expected Outcome	Actual Outcome	Corrective action	Page of Evidence
Process [A]:		Information Display					
2.6[A][1]	Instrument Information	To ensure that all information is relevant to the selected student.	N/A	All details should correlate to the Instrument selected by the user.	As Expected.	N/A	
Process [B]:		Record Navigation					
2.6[B][1]	Instrument Search	To ensure that the search function is operational.	A combo box detailing the name for each Instrument records.	Upon selecting a value present within the search combo box, the form display should update, to display the associated details.	As Expected.	N/A	
2.6[B][2]	Next Instrument	To ensure that the button operates as intended.	(Selected Instrument ID + 1)	Selecting this button should cause the form display to update, as to detail	As Expected.	N/A	

				information regarding the next Instrument present within the table.			
2.6[B][3]	Previous Instrument	To ensure that the button operates as intended.	(Selected Instrument ID – 1)	Selecting this button should cause the form display to update, as to detail information regarding the previous Instrument present within the table.	As Expected.	N/A	
2.6[B][4]	First Instrument	To ensure that the button operates as intended.	(Selected Instrument ID = AllInstrument IDs[0])	Selecting this button should cause the form display to update, as to detail information regarding the first Instrument record present within the	As Expected.	N/A	

				table.			
2.6[B][5]	Last Instrument	To ensure that the button operates as intended.	(Selected Instrument ID = (AllInstrument. Count() – 1))	Selecting this button should cause the form display to update, as to detail information regarding the last Instrument record present within the table.	As Expected.	N/A	
Process [C]:		Form Navigation					
2.6[C][1]	Add New Instrument Record	To ensure that the button operates as intended.	Global.Previous Form = “InstrumentForm _Add”.	Selecting this button should cause this form to close, and the Add Field Form to appear.	As Expected.	N/A	
2.6[C][2]	Update Instrument Record	To ensure that the button operates as intended.	Global.Previous Form = “InstrumentForm _Update”.	Selecting this button should cause this form to close, and the Add Field Form to appear.	A simple problem that was identified in association with this button, was that no	This was simply due to the fact ta I had forgotten to add the Mouse Over, and Mouse Out	

					feedback was provided when hovering the mouse over.	events for this button, and therefore the necessary code.	
2.6[C][3]	Delete Instrument Record [User Logged in]	To ensure that the button operates as intended.	Global.Previous Form = "InstrumentForm". Global.ID = (Selected Instrument ID) Global.FieldName = (Selected Name)	Selecting this button should cause this form to close, and the Confirmation Form to appear.	As Expected.	N/A	
2.6[C][4]	Delete Instrument Record [No user logged in]	To ensure that the button operates as intended.	N/A	Selecting this button should cause this form to close, and the Login Form to appear.	As Expected.	N/A	
2.6[C][5]	Return to Main Menu	To ensure that the button operates as intended.	N/A	Selecting this button should cause this form to close, and the Private Tuition	As Expected.	N/A	

Centre Number: 71348

Candidate Number: 0075

				Form to appear.				
--	--	--	--	------------------------	--	--	--	--

2.7) Grade Form

frmSplash								
Test Number	Tested Data	Test Purpose	Additional Details	Expected Outcome	Actual Outcome	Corrective action	Page of Evidence	
Process [A]:		Information Display						
2.7[A][1]	Grade Information	To ensure that all information is relevant to the selected student.	N/A	All details should correlate to the Grade selected by the user.	As Expected.	N/A		
Process [B]:		Record Navigation						
2.7[B][1]	Grade Search	To ensure that the search function is operational.	A combo box detailing the Grade Levels for all records.	Upon selecting a value present within the search combo box, the form display should update, to display the associated details.	As Expected.	N/A		
2.7[B][2]	Next Grade	To ensure that the button operates as intended.	(Selected Grade ID + 1)	Selecting this button should cause the form display to update, as to detail	As Expected.	N/A		

				information regarding the next Grade present within the table.			
2.7[B][3]	Previous Grade	To ensure that the button operates as intended.	(Selected Grade ID – 1)	Selecting this button should cause the form display to update, as to detail information regarding the previous Grade present within the table.	As Expected.	N/A	
2.7[B][4]	First Grade	To ensure that the button operates as intended.	(Selected Grade ID = AllGradeIDs[0])	Selecting this button should cause the form display to update, as to detail information regarding the first Grade record present within the table.	As Expected.	N/A	

2.7[B][5]	Last Grade	To ensure that the button operates as intended.	(Selected Grade ID = (AllGradeIDs. Count() – 1))	Selecting this button should cause the form display to update, as to detail information regarding the last Grade record present within the table.	As Expected.	N/A	
Process [C]:		Form Navigation					
2.7[C][1]	Add New Grade Record	To ensure that the button operates as intended.	Global.Previous Form = “GradeTable _Add”.	Selecting this button should cause this form to close, and the Add Field Form to appear.	As Expected.	N/A	
2.7[C][2]	Update Grade Record	To ensure that the button operates as intended.	Global.Previous Form = “GradeTable _Update”.	Selecting this button should cause this form to close, and the Add Field Form to appear.	All record information was displayed correctly within the form, bar the Grade Fee value.	Upon reviewing the system, it was identified that the source of the problem was a simple mistake made in relation to the information	

						display. Previous intentions for this field related to the belief that the information would be displayed and selected from a combo box. It would seem that while this idea was scrapped, I had forgotten to replace the code.	
2.7[C][3]	Delete Grade Record [User Logged in]	To ensure that the button operates as intended.	Global.Previous Form = "GradeTable". Global.ID = (Selected Grade ID)	Selecting this button should cause this form to close, and the Confirmation Form to appear.	As Expected.	N/A	

			Global.FieldName = (Selected GradeLevel)				
2.7[C][4]	Delete Grade Record [No user logged in]	To ensure that the button operates as intended.	N/A	Selecting this button should cause this form to close, and the Login Form to appear.	As Expected.	N/A	
2.7[C][5]	Return to Main Menu	To ensure that the button operates as intended.	N/A	Selecting this button should cause this form to close, and the Private Tuition Form to appear.	As Expected.	N/A	

2.8) Lesson Bundle Form

frmSplash							
Test Number	Tested Data	Test Purpose	Additional Details	Expected Outcome	Actual Outcome	Corrective action	Page of Evidence
Process [A]:		Information Display					
2.8[A][1]	Lesson Bundle Information	To ensure that all information is relevant to the selected student.	N/A	All details should correlate to the Lesson Bundle selected by the user.	As Expected.	N/A	
Process [B]:		Record Navigation					
2.8[B][1]	Lesson Bundle Search	To ensure that the search function is operational.	A combo box detailing the Bundle Name for each Lesson Bundle Record.	Upon selecting a value present within the search combo box, the form display should update, to display the associated details.	As Expected.	N/A	
2.8[B][2]	Next Lesson Bundle	To ensure that the button operates as intended.	(Selected LessonBundle ID + 1)	Selecting this button should cause the form display to update, as to detail	As Expected.	N/A	

				information regarding the next student present within the table.			
2.8[B][3]	Previous Lesson Bundle	To ensure that the button operates as intended.	(Selected Lesson Bundle ID – 1)	Selecting this button should cause the form display to update, as to detail information regarding the previous Lesson Bundle present within the table.	As Expected.	N/A	
2.8[B][4]	First Lesson Bundle	To ensure that the button operates as intended.	(Selected Lesson Bundle ID = AllLesson BundleIDs[0])	Selecting this button should cause the form display to update, as to detail information regarding the first Lesson Bundle record present within the table.	As Expected.	N/A	

2.8[B][5]	Last Lesson Bundle	To ensure that the button operates as intended.	(Selected Lesson Bundle ID = (AllLesson BundleIDs. Count() – 1))	Selecting this button should cause the form display to update, as to detail information regarding the last Lesson Bundle record present within the table.	As Expected.	N/A	
Process [C]:		Form Navigation					
2.8[C][1]	Add New Lesson Bundle Record	To ensure that the button operates as intended.	Global.Previous Form = “Lesson BundleTable _Add”.	Selecting this button should cause this form to close, and the Add Field Form to appear.	As Expected.	N/A	
2.8[C][2]	Update Lesson Bundle Record	To ensure that the button operates as intended.	Global.Previous Form = “Lesson BundleTable _Update”.	Selecting this button should cause this form to close, and the Add Field Form to appear.	As Expected.	N/A	
2.8[C][3]	Delete Lesson Bundle Record [User Logged	To ensure that the button operates as	Global.Previous Form = “Lesson BundleTable”.	Selecting this button should cause this form to	As Expected.	N/A	

	in]	intended.	Global.ID = (Selected Lesson Bundle ID) Global.FieldName = (Selected BundleName)	close, and the Confirmation Form to appear.			
2.8[C][4]	Delete Lesson Bundle Record [No user logged in]	To ensure that the button operates as intended.	N/A	Selecting this button should cause this form to close, and the Login Form to appear.	As Expected.	N/A	
2.8[C][5]	Return to Main Menu	To ensure that the button operates as intended.	N/A	Selecting this button should cause this form to close, and the Private Tuition Form to appear.	As Expected.	N/A	

2.9) Room Form

frmRoomTable								
Test Number	Tested Data	Test Purpose	Additional Details	Expected Outcome	Actual Outcome	Corrective action	Page of Evidence	
Process [A]:		Information Display						
2.9[A][1]	Room Information	To ensure that all information is relevant to the selected student.	N/A	All details should correlate to the Room selected by the user.	As Expected.	N/A		
Process [B]:		Record Navigation						
2.9[B][1]	Room Search	To ensure that the search function is operational.	A combo box detailing the Number for each Room Record.	Upon selecting a value present within the search combo box, the form display should update, to display the associated details.	As Expected.	N/A		
2.9[B][2]	Next Room	To ensure that the button operates as intended.	(Selected Room ID + 1)	Selecting this button should cause the form display to update, as to detail	As Expected.	N/A		

				information regarding the next Room present within the table.			
2.9[B][3]	Previous Room	To ensure that the button operates as intended.	(Selected Room ID – 1)	Selecting this button should cause the form display to update, as to detail information regarding the previous Room present within the table.	As Expected.	N/A	
2.9[B][4]	First Room	To ensure that the button operates as intended.	(Selected Room ID = AllRoomIDs[0])	Selecting this button should cause the form display to update, as to detail information regarding the first Room record present within the table.	As Expected.	N/A	

2.9[B][5]	Last Room	To ensure that the button operates as intended.	(Selected Room ID = (AllRoomIDs. Count() – 1))	Selecting this button should cause the form display to update, as to detail information regarding the last Room record present within the table.	As Expected.	N/A	
Process [C]:		Form Navigation					
2.9[C][1]	Add New Room Record	To ensure that the button operates as intended.	Global.Previous Form = “RoomTable _Add”.	Selecting this button should cause this form to close, and the Add Field Form to appear.	As Expected.	N/A	
2.9[C][2]	Update Room Record	To ensure that the button operates as intended.	Global.Previous Form = “RoomTable _Update”.	Selecting this button should cause this form to close, and the Add Field Form to appear.	As Expected.	N/A	
2.9[C][3]	Delete Room Record	To ensure that the button	Global.Previous Form =	Selecting this button should	As Expected.	N/A	

	[User Logged in]	operates as intended.	"RoomTable". Global.ID = (Selected Room ID) Global.FieldName = (Selected RoomNumber)	cause this form to close, and the Confirmation Form to appear.			
2.9[C][4]	Delete Room Record [No user logged in]	To ensure that the button operates as intended.	N/A	Selecting this button should cause this form to close, and the Login Form to appear.	As Expected.	N/A	
2.9[C][5]	Return to Main Menu	To ensure that the button operates as intended.	N/A	Selecting this button should cause this form to close, and the Private Tuition Form to appear.	As Expected.	N/A	

2.10) Purchased Lesson Bundles Form

frmPurchasedLessons							
Test Number	Tested Data	Test Purpose	Additional Details	Expected Outcome	Actual Outcome	Corrective action	Page of Evidence
Process [A]:		Information Display					
2.10[A][1]	Student Information	To ensure that all information is relevant to the selected student.	N/A	All details should correlate to the student selected by the user.	As Expected.	N/A	
2.10[A][2]	Grade Information	To ensure that all information is relevant to the selected student.	Displays information in relation to the Grade ID present within the student information.	All information present within the display should correlate with the selected student record.	As Expected.	N/A	
2.10[A][3]	Lesson Bundle Information	To ensure that all information is relevant to the selected Lesson.	Displays information in relation to the Bundle ID present within the selected Purchased Lessons information.	All information present within the display should correlate with the selected lesson record.	As Expected.	N/A	

2.10[A][4]	Purchased Lessons Information	To ensure that all information is relevant to the selected Student.	Will display information in relations to the selected student, and their associated ID.	All information present within this display should correlate with the selected student ID.	As Expected.	N/A	
Process: [B]:		Record Navigation					
2.10[B][1]	Student Search	To ensure that the search function is operational.	A combo box detailing the first and last name for all students is present within the form.	Upon selecting a value present within the search combo box, the form display should update, to display the associated details.	As Expected.	N/A	
2.10[B][2]	Next Student	To ensure that the button operates as intended.	(Selected Student ID + 1)	Selecting this button should cause the form display to update, as to detail information regarding the next student present within the table.	As Expected.	N/A	

2.10[B][3]	Previous Student	To ensure that the button operates as intended.	(Selected Student ID – 1)	Selecting this button should cause the form display to update, as to detail information regarding the previous student present within the table.	As Expected.	N/A	
2.10[B][4]	First Student	To ensure that the button operates as intended.	(Selected Student ID = AllStudentIDs[0])	Selecting this button should cause the form display to update, as to detail information regarding the first student record present within the table.	As Expected.	N/A	
2.10[B][5]	Last Student	To ensure that the button operates as intended.	(Selected Student ID = (AllStudentIDs.Count() – 1))	Selecting this button should cause the form display to update, as to detail	As Expected.	N/A	

				information regarding the last student record present within the table.			
Process [C]:		Form Navigation					
2.10[C][1]	Add New Purchased Lesson Record	To ensure that the button operates as intended.	Global.Previous Form = "PurchasedLessons Table_Add".	Selecting this button should cause this form to close, and the Add Field Form to appear.	As Expected.	N/A	
2.10[C][2]	Update Purchased Lesson Record	To ensure that the button operates as intended.	Global.Previous Form = "PurchasedLessons Table_Update".	Selecting this button should cause this form to close, and the Add Field Form to appear.	Upon selecting the update button, a blank Add Field form was loaded. There were a few different reasons for this display error, the first of which being that I had been passing an incorrect value, one which is used to identify the display properties, and assign them to	To solve these problems, the mentioned code was simply added to the button. This simply means that the correct display value is assigned, and that the necessary information is	

					<p>the form.</p> <p>In addition to this, I realised that there were additional lines of code that were missing in association to this button. This included the storage of information, and then passing that associated with the selected record through.</p>	<p>being transferred.</p>	
2.10[C][3]	Delete Purchased Lesson Record [User Logged in]	To ensure that the button operates as intended.	<p>Global.Previous Form = "PurchasedLessons Table".</p> <p>Global.ID = (Purchased Lesson ID)</p> <p>Global.FieldName = (Selected First Name + Surname)</p>	Selecting this button should cause this form to close, and the Confirmation Form to appear.	As Expected.	N/A	

2.10[C][4]	Delete Purchased Lesson Record [No user logged in]	To ensure that the button operates as intended.	N/A	Selecting this button should cause this form to close, and the Login Form to appear.	As Expected.	N/A	
2.10[C][5]	Return to Main Menu	To ensure that the button operates as intended.	N/A	Selecting this button should cause this form to close, and the Private Tuition Form to appear.	As Expected.	N/A	

2.11) Scheduled Lessons Form

frmScheduleTable							
Test Number	Tested Data	Test Purpose	Additional Details	Expected Outcome	Actual Outcome	Corrective action	Page of Evidence
Process [A]:		Information Display					
2.11[A][1]	Student Information	To ensure that all information is relevant to the selected student.	N/A	All details should correlate to the student selected by the user.	As Expected.	N/A	
2.11[A][2]	Grade Information	To ensure that all information is relevant to the selected student.	Identifies information in correspondence with the selected student ID.	Displays should contain Grade information which relates to the selected student ID.	As Expected.	N/A	
2.11[A][3]	Teacher Information	To ensure that all information is relevant to the selected Scheduled Lesson.	Identifies information in correspondence with selected Scheduled Lesson.	Displays should contain Teacher information in association to the selected scheduled lesson.	As Expected.	N/A	

2.11[A][4]	Scheduled Lessons Information	To ensure that all information is relevant to the selected Scheduled Lesson.	Identifies information in correspondence with the selected student ID.	Displays should contain Grade information which relates to the selected student ID.	As Expected.	N/A	
Process: [B]:		Record Navigation					
2.11[B][1]	Student Search	To ensure that the search function is operational.	A combo box detailing the first and last name for all students is present within the form.	Upon selecting a value present within the search combo box, the form display should update, to display the associated details.	As Expected.	N/A	
2.11[B][2]	Next Student	To ensure that the button operates as intended.	(Selected Student ID + 1)	Selecting this button should cause the form display to update, as to detail information regarding the next student present within the table.	As Expected.	N/A	

2.11[B][3]	Previous Student	To ensure that the button operates as intended.	(Selected Student ID – 1)	Selecting this button should cause the form display to update, as to detail information regarding the previous student present within the table.	As Expected.	N/A	
2.11[B][4]	First Student	To ensure that the button operates as intended.	(Selected Student ID = AllStudentIDs[0])	Selecting this button should cause the form display to update, as to detail information regarding the first student record present within the table.	As Expected.	N/A	
2.11[B][5]	Last Student	To ensure that the button operates as intended.	(Selected Student ID = (AllStudentIDs.Count() – 1))	Selecting this button should cause the form display to update, as to detail	As Expected.	N/A	

				information regarding the last student record present within the table.			
Process [C]:		Form Navigation					
2.11[C][1]	Add New Scheduled Lesson Record	To ensure that the button operates as intended.	Global.Previous Form = "ScheduleTable". Purpose = "Add"	Selecting this button should cause this form to close, and the Add Field Form to appear.	As Expected.	N/A	
2.11[C][2]	Update Scheduled Lesson Record (No Record Selected)	To ensure that the button operates as intended.	Global.Previous Form = "ScheduleTable". Purpose = "Update"	Selecting this button should cause this form to close, and the Add Field Form to appear.	As Expected.	N/A	
2.11[C][3]	Update Scheduled Lesson Record (Record Selected)	To ensure that the button operates as intended.	Global.Previous Form = "ScheduleTable". Purpose = "Update"	Selecting this button should cause this form to close, and the Add Field form to appear.	As Expected.	N/A	

2.11[C][4]	Delete Scheduled Lesson Record [User Logged in] (No Record Selected)	To ensure that the button operates as intended.	<pre>Global.PreviousForm = "ScheduleTable". Global.ID = (Selected ScheduledLesson ID) Global.FieldName = (Selected First Name + Surname)</pre>	Selecting this button should cause this form to close, and the Confirmation Form to appear.	As Expected.	N/A	
2.11[C][5]	Delete Scheduled Lesson Record [User Logged in] (Record Selected)	To ensure that the button operates as intended.	<pre>Global.PreviousForm = "ScheduleTable". Global.ID = (Selected ScheduledLesson ID) Global.FieldName = (Selected First Name + Surname)</pre>	Selecting this button should cause this form to close, and the Confirmation Form to appear.	Upon selecting this button, the system crashed. It was discovered that this was likely due to the fact that there were no constraints set in regards to the user passing a null scheduled record.	To solve this, a simple if statement was implemented, as to ensure that this code would only run in the event that the user has selected a record. Otherwise an error message will appear.	

2.11[C][6]	Delete Scheduled Lesson Record [No user logged in]	To ensure that the button operates as intended.	N/A	Selecting this button should cause this form to close, and the Login Form to appear.	While testing these conditions, it was discovered that the system would crash. This was due to the fact that the three were no constraints as to prevent the system from passing through a Null value.	A simple fix to this problem was to establish an if statement which would cause an error message to appear should not record be selected.	
2.11[C][7]	Return to Main Menu	To ensure that the button operates as intended.	N/A	Selecting this button should cause this form to close, and the Private Tuition Form to appear.	As Expected.	N/A	
2.11[C][8]	Return to Calendar	To ensure that the button operates as intended.	If user has accessed this form through the calendar, then this button will be	Selecting this button should cause this form to close, and the calendar form to	As Expected.	N/A	

			visible.	appear.				
--	--	--	----------	---------	--	--	--	--

2.12) Archived Scheduled Lessons

frmArchive_Schedule							
Test Number	Tested Data	Test Purpose	Additional Details	Expected Outcome	Actual Outcome	Corrective action	Page of Evidence
Process [A]:		Information Display					
2.12[A][1]	Student Information	To ensure that all information is relevant to the selected student.	N/A	All details should correlate to the student selected by the user.	As Expected.	N/A	
2.12[A][2]	Grade Information	To ensure that all information is relevant to the selected student.	Identifies information in correspondence with the selected student ID.	Displays should contain Grade information which relates to the selected student ID.	As Expected.	N/A	
2.12[A][3]	Teacher Information	To ensure that all information is relevant to the selected Scheduled Lesson.	Identifies information in correspondence with selected Scheduled Lesson.	Displays should contain Teacher information in association to the selected scheduled lesson.	As Expected.	N/A	

2.12[A][4]	Scheduled Lessons Information	To ensure that all information is relevant to the selected Scheduled Lesson.	Identifies information in correspondence with the selected student ID.	Displays should contain Grade information which relates to the selected student ID.	As Expected.	N/A	
Process: [B]:		Record Navigation					
2.12[B][1]	Student Search	To ensure that the search function is operational.	A combo box detailing the first and last name for all students is present within the form.	Upon selecting a value present within the search combo box, the form display should update, to display the associated details.	As Expected.	N/A	
2.12[B][2]	Next Student	To ensure that the button operates as intended.	(Selected Student ID + 1)	Selecting this button should cause the form display to update, as to detail information regarding the next student present within the table.	As Expected.	N/A	

2.12[B][3]	Previous Student	To ensure that the button operates as intended.	(Selected Student ID – 1)	Selecting this button should cause the form display to update, as to detail information regarding the previous student present within the table.	As Expected.	N/A	
2.12[B][4]	First Student	To ensure that the button operates as intended.	(Selected Student ID = AllStudentIDs[0])	Selecting this button should cause the form display to update, as to detail information regarding the first student record present within the table.	As Expected.	N/A	
2.12[B][5]	Last Student	To ensure that the button operates as intended.	(Selected Student ID = (AllStudentIDs.Count() – 1))	Selecting this button should cause the form display to update, as to detail	As Expected.	N/A	

				information regarding the last student record present within the table.			
Process [C]:		Form Navigation					
2.12[C][3]	Delete Scheduled Lesson Record [User Logged in]	To ensure that the button operates as intended.	Global.Previous Form = "Archive_ScheduleTable". Global.ID = (Selected Archive_ScheduleTable ID) Global.FieldName = (Selected First Name + Surname)	Selecting this button should cause this form to close, and the Confirmation Form to appear.	As Expected.	N/A	
2.12[C][4]	Delete Scheduled Lesson Record [No user logged in]	To ensure that the button operates as intended.	N/A	Selecting this button should cause this form to close, and the Login Form to appear.	As Expected.	N/A	

2.12[C][6]	Return to Main Menu	To ensure that the button operates as intended.	N/A	Selecting this button should cause this form to close, and the Private Tuition Form to appear.	As Expected.	N/A	
-------------------	---------------------	---	-----	--	--------------	-----	--

2.13) Reallocated Students

frmStudentTable								
Test Number	Tested Data	Test Purpose	Additional Details	Expected Outcome	Actual Outcome	Corrective action	Page of Evidence	
Process [A]:		Information Display						
2.13[A][1]	Student Information	To ensure that all information is relevant to the selected student.	N/A	All details should correlate to the student selected by the user.	As Expected.	N/A		
Process [B]:		Record Navigation						
2.13[B][1]	Student Search	To ensure that the search function is operational.	A combo box detailing the first and last name for all students is present within the form.	Upon selecting a value present within the search combo box, the form display should update, to display the associated details.	As Expected.	N/A		
2.13[B][2]	Next Student	To ensure that the button operates as intended.	(Selected Student ID + 1)	Selecting this button should cause the form display to update, as to detail	As Expected.	N/A		

				information regarding the next student present within the table.			
2.13[B][3]	Previous Student	To ensure that the button operates as intended.	(Selected Student ID – 1)	Selecting this button should cause the form display to update, as to detail information regarding the previous student present within the table.	As Expected.	N/A	
2.13[B][4]	First Student	To ensure that the button operates as intended.	(Selected Student ID = AllStudentIDs[0])	Selecting this button should cause the form display to update, as to detail information regarding the first student record present within the table.	As Expected.	N/A	

2.13[B][5]	Last Student	To ensure that the button operates as intended.	(Selected Student ID = (AllStudentIDs. Count() – 1))	Selecting this button should cause the form display to update, as to detail information regarding the last student record present within the table.	As Expected.	N/A	
Process [C]:		Form Navigation					
2.13[C][3]	Delete Student Record [User Logged in]	To ensure that the button operates as intended.	Global.ID = (Selected Reallocated_ Student ID) Global.FieldName = (Selected FirstName + Surname)	Selecting this button should cause this form to close, and the Confirmation Form to appear.	As Expected.	N/A	
2.13[C][4]	Delete Student Record [No user logged in]	To ensure that the button operates as intended.	N/A	Selecting this button should cause this form to close, and the Login Form to appear.	As Expected.	N/A	

2.13[C][5]	Return to Main Menu	To ensure that the button operates as intended.	N/A	Selecting this button should cause this form to close, and the Private Tuition Form to appear.	As Expected.	N/A	
-------------------	---------------------	---	-----	--	--------------	-----	--

2.14) Schedule Calendar - [Date]

frmCalendarDate:							
Test Number	Tested Data	Test Purpose	Additional Details	Expected Outcome	Actual Outcome	Corrective action	Page of Evidence
Process [A]:		Calendar					
2.14[A][1]	Calendar Navigation {Select Arrows}	To ensure that calendar navigation is operational.	Calendar Displays two months at a time.	Selecting these buttons, should cause the next, or previous 2 months to appear within the display, in correspondence to the selected button.	As Expected.	N/A	
2.14[A][2]	Calendar Error {Select Weekend}	To ensure that invalid dates prompt error messages.	If Selected Date == “Saturday” or “Sunday”: Display Error Message(“Weekends are reserved for specialised classes”)	Selecting a weekend date, should cause an error message to appear.	As Expected.	N/A	
2.14[A][3]	Calendar Error {Select Summer Date}	To ensure that invalid dates prompt error	If Selected Month == “07” or “08”: Display Error	Selecting a date present within the Summer months of	As Expected.	N/A	

		messages.	Message("Private classes are unavailable during Summer Months")	August July or August should cause an error message to appear.			
2.14[A][4]	Calendar {Select Valid Date}	To ensure that selecting a valid date, causes a form transition.	Store selected Date values.	Selecting a valid date should cause this form to close, and the Calendar Times form to appear.	As Expected.	N/A	
Process: [B]:	Search Function						
2.14[B][1]	Search by Month	To ensure that this search function is operational.	A combo box is present within the form, and will detail a list of all months within the year.	Upon selecting one of the months, the calendar display should update, as to display the selected month.	As Expected.	N/A	
2.14[A][2]	Calendar Navigation {Select Date Display}	To ensure that calendar navigation is operational.	N/A	Selecting this button should cause a new display to appear, detailing a list of all months. This	As Expected.	N/A	

				could be used as an alternative search function.			
Process [C]:		Form Navigation Buttons:					
2.14[C][1]	Return to Main Menu	To ensure that the selected button completes the intended operation.	N/A	Selecting this button should cause this form to close, and the Private Tuition form to appear.	As Expected.	N/A	

2.15) Schedule Calendar [Time]

frmCalendarTime:								
Test Number	Tested Data	Test Purpose	Additional Details	Expected Outcome	Actual Outcome	Corrective action	Page of Evidence	
Process [A]:		Form Display						
2.15[A][1]	Date Display	To ensure that the correct information is being displayed within the form.		This display should update, as to display information which corresponds to that selected within the previous form.	As Expected.	N/A		
Process [B]:		Time Selection						
2.15[B][1]	Time Based buttons	To ensure that the intended operations run when these buttons are selected.	Stores Selected time.	Selecting any of these buttons should cause this form to close, and prompt the classSchedule form to appear.	While conducting this test, it was acknowledged that selecting a time value would cause the system to crash. This was due to the fact that the data type for the time field present within	To solve this problem, I simply had to modify the data reader, so that it would collect a string instead of a time value, and the also update the associated lists.		

					the schedule table had been modified.		
Process [C]:		Form Navigation					
2.5[C][1]	Return to Calendar	To ensure that the intended operation runs when this button is selected.		Selecting this button should cause this form to close, and the CalendarDates form to appear.	As Expected.	N/A	

2.16) Schedule Calendar [Scheduled Lessons]

frmClassSchedule							
Test Number	Tested Data	Test Purpose	Additional Details	Expected Outcome	Actual Outcome	Corrective action	Page of Evidence
Process [A]:		Form Display					
2.16[A][1]	Date Display	To ensure that the correct information is displayed.		This display should detail the selected date from one of the previous form.	As Expected.	N/A	
2.16[A][2]	Time Display	To ensure that the correct information is displayed.		This display should detail the selected time from the previous form.	As Expected.	N/A	
2.16[A][3]	Schedule Information	To ensure that the correct information is displayed.		This information should relate to the selected time and date.	As Expected.	N/A	
2.16[A][4]	Error Message	To ensure that the correct information is displayed.	IF no scheduled lessons: There are currently no scheduled classes for this date and	In the event that there are no scheduled lessons for the selected time and date, then an error	As Expected.	N/A	

			time."	message should be displayed.			
2.16[A][5]	Information Display Panels	To ensure that the correct information is displayed.		This form contains 6 panels, in the event that there are less than 6 scheduled lessons for the selected date and time. These panels should be hidden accordingly.	As Expected.	N/A	
Process: [B]:		Form Navigation					
2.16[B][1]	Return to Calendar	To ensure that the specified operation runs, when this button is selected.		Selecting this button should cause this form to close, and the calendarTimes from should appear.	As Expected.	N/A	
2.16[B][2]	Return to Main Menu	To ensure that the specified operation runs, when this button is selected.		Selecting this button should cause this form to close, and prompt the Private Tuition form to	As Expected.	N/A	

				appear.			
2.16[B][3]	Schedule New Lesson	To ensure that the specified operation runs, when this button is selected.	Store Previous form value as "ScheduleTable _Add".	Selecting this button should cause this form to close, and then prompt the Add Field form to appear.	As Expected.	N/A	
2.16[B][4]	Information Display Panel	To ensure that the specified operation runs, when this object is selected.	Stores associated schedule ID.	Selecting this object should cause this form to close, and prompt the scheduledTable form to appear.	As Expected.	N/A	

2.17) Add New Field**2.17[i]) Student Record****2.17[i][A]) Student Validation**

frmAddField							
Test Number	Tested Data	Test Purpose	Additional Details	Expected Outcome	Actual Outcome	Corrective action	Page of Evidence
Category [i]							
Process [A]:							
2.17[i][A] [1]	First Name Validation	To ensure that the validation code is operational, and thus prevents or at least limits the instances for which users enter invalid characters.	{Textbox is blank}	An error message should appear, stating that the user must enter a value into the associated textbox.	As Expected.	N/A	
			{Numeric Character Entered}	An error message should appear, requesting that the user remove the numeric character.	As Expected.	N/A	
			{Symbolic Character Entered}	An error message should appear, requesting that the user remove	As Expected.	N/A	

				the Symbol character.			
2.17[i][A] [2]	Other Name(s) Validation	To ensure that the validation code is operational, and thus prevents or at least limits the instances for which users enter invalid characters.	{Punctual Character Entered}	An error message should appear, requesting that the user remove the punctuation character.	As Expected.	N/A	
			Text box max Length = 50;	The user should be restricted from entering more than 50 characters into this object.	As Expected.	N/A	
			{Textbox is blank}	An error message should appear, stating that the user must enter a value into the associated textbox.	As Expected.	N/A	
			{Numeric Character Entered}	An error message should appear, requesting that the user remove	As Expected.	N/A	

				the numeric character.			
			{Symbolic Character Entered}	An error message should appear, requesting that the user remove the Symbol character.	As Expected.	N/A	
			{Punctual Character Entered}	An error message should appear, requesting that the user remove the punctuation character.	As Expected.	N/A	
			Text box max Length = 50;	The user should be restricted from entering more than 50 characters into this object.	As Expected.	N/A	
2.17[i][A] [B]	Surname Validation	To ensure that the validation code is operational, and thus	{Textbox is blank}	An error message should appear, stating that the user must enter a value into the	As Expected.	N/A	

		prevents or at least limits the instances for which users enter invalid characters.		associated textbox.			
		{Numeric Character Entered}		An error message should appear, requesting that the user remove the numeric character.	As Expected.	N/A	
		{Symbolic Character Entered}		An error message should appear, requesting that the user remove the Symbol character.	As Expected.	N/A	
		{Punctual Character Entered}		An error message should appear, requesting that the user remove the punctuation character.	As Expected.	N/A	
		Text box max Length = 50;		The user should be restricted from entering more than 50 characters into	As Expected.	N/A	

				this object.			
2.17[i][A] [4]	Date Of Birth Validation	To ensure that the validation code is operational, and thus prevents or at least limits the instances for which users enter invalid characters.	{Textbox is blank}	An error message should appear, stating that the user must enter a value into the associated textbox.	As Expected.	N/A	
2.17[i][A] [5]	Address Validation	To ensure that the validation code is operational, and thus prevents or at least limits the instances for which users enter invalid characters.	{Textbox is blank}	An error message should appear, stating that the user must enter a value into the associated textbox.	As Expected.	N/A	

			{Symbolic Character Entered}	An error message should appear, requesting that the user remove the Symbol character.	As Expected.	N/A	
			{Punctual Character Entered}	An error message should appear, requesting that the user remove the punctuation character.	As Expected.	N/A	
			Text box max Length = 50;	The user should be restricted from entering more than 50 characters into this object.	As Expected.	N/A	
2.17[i][A] [6]	Town Validation	To ensure that the validation code is operational, and thus prevents or at least limits the instances for	{Textbox is blank}	An error message should appear, stating that the user must enter a value into the associated textbox.	As Expected.	N/A	

		which users enter invalid characters.	{Numeric Character Entered}	An error message should appear, requesting that the user remove the numeric character.	As Expected.	N/A	
			{Symbolic Character Entered}	An error message should appear, requesting that the user remove the Symbol character.	As Expected.	N/A	
			{Punctual Character Entered}	An error message should appear, requesting that the user remove the punctuation character.	As Expected.	N/A	
			Text box max Length = 50;	The user should be restricted from entering more than 50 characters into this object.	As Expected.	N/A	

2.17[i][A] [7]	Post Code Validation	<p>To ensure that the validation code is operational, and thus prevents or at least limits the instances for which users enter invalid characters.</p>	{Textbox is blank}	An error message should appear, stating that the user must enter a value into the associated textbox.	As Expected.	N/A	
			[First two Characters = BT]	An error message should appear, stating that the first two characters must be "BT".	As Expected.	N/A	
			[Next Three Characters – Letters] {Numeric Character Present}	An error message should appear, stating that there is a numerical character present within the specified character range, and must be removed.	As Expected.	N/A	
			[Next Three Characters – Letters]	An error message should appear, stating that there	As Expected.	N/A	

			{Symbol Character Present}	is a symbol character present within the specified character range, and must be removed.			
			[Next Three Characters – Letters] {Punctual Character Present}	An error message should appear, stating that there is a punctual character present within the specified character range, and must be removed.	As Expected.	N/A	
			[Final Three Characters – Numbers] {Alphabetical Character Present}	An error message should appear, stating that there is a Alphabetical character present within the specified character range, and must be removed.	As Expected.	N/A	

			[Final Three Characters – Numbers] {Symbol Character Present}	An error message should appear, stating that there is a symbol character present within the specified character range, and must be removed.	As Expected.	N/A	
			[Final Three Characters – Numbers] {Punctual Character Present}	An error message should appear, stating that there is a punctual character present within the specified character range, and must be removed.	As Expected.	N/A	
			{8 Characters Present}	Should the textbox contain less than 8 characters. An error message should appear, detain the format	As Expected.	N/A	

				of the desired Post Code.			
2.17[i][A] [8]	Contact Number Validation	To ensure that the validation code is operational, and thus prevents or at least limits the instances for which users enter invalid characters.	{No value Present}	An error message should appear, stating that a value must be present.	As Expected.	N/A	
			{Letter Present}	An error message should appear, prompting the user to remove the alphabetical character.	As Expected.	N/A	
			{Symbol Present}	An error message should appear, prompting the user to remove the symbol character.	As Expected.	N/A	
			{Punctuation Present}	An error message should appear, prompting the user to remove the punctuation character.	As Expected.	N/A	

2.17[i][A] [9]	Email Address Validation	<p>To ensure that the validation code is operational, and thus prevents or at least limits the instances for which users enter invalid characters.</p>	{No Value Present}	An error message should appear, stating that this field must be completed.	As Expected.	N/A	
			{No "@" Present}	An error message should appear, stating that the Email Address value must contain a "@" symbol.	As Expected.	N/A	
			{Multiple "@"s Present}	An error message should appear, stating that an Email Address cannot contain multiple "@" symbols.	As Expected.	N/A	
			Text box max Length = 100;	The user should be restricted from entering more than 50 characters into this object.	As Expected.	N/A	

2.17[i][A] [10]	Grade ID Validation	To ensure that the validation code is operational, and thus prevents or at least limits the instances for which users enter invalid characters.	{User Must Select Grade Value}	A combo box should ensure that the user must select a value. In addition to this, it should prevent the user from entering a custom value.	As Expected.	N/A	
2.17[i][A] [11]	Instrument Validation	To ensure that the validation code is operational, and thus prevents or at least limits the instances for which users enter invalid characters.	{User Must Select Instrument Value}	A combo box should ensure that the user must select a value. In addition to this, it should prevent the user from entering a custom value.	As Expected.	N/A	
2.17[i][A] [12]	Tuition Fee	To ensure that the validation code is operational, and thus	{User must possess the ability to tick tuition fee}	A check box should provide the user with the ability to tick and un-tick the	As Expected.	N/A	

		prevents or at least limits the instances for which users enter invalid characters.		component as to represent whether or not the new student has paid their tuition fee.			
--	--	---	--	--	--	--	--

2.17[i][B]) Student Update Display

Test Number	Tested Data	Test Purpose	Additional Details	Expected Outcome	Actual Outcome	Corrective action	Page of Evidence
Category [i]		Student Record					
Process [B]:		Update Display					
2.17[ii][B] [1]	Student Information Display	To ensure that the correct information is displayed	N/A	Upon selecting the student update button, the details associated with the selected student record should appear within the form.	As Expected.	N/A	

2.17[i][C] Student Modifications

Test Number	Tested Data	Test Purpose	Additional Details	Expected Outcome	Actual Outcome	Corrective action	Page of Evidence
Category [i]		Student Record					
Process [C]:		Data Modifications					
2.17[i][C] [1]	Add New Student Record	To ensure that this operation is functioning as intended.	N/A	Should the user enter valid information for all student fields, upon selecting the button the student should be added to the table.	As Expected.	N/A	
2.17[i][C] [2]	Update Student Record	To ensure that this operation is functioning as intended.	N/A	Should the user enter valid information for all student fields present within the form, upon selecting the button the student record should be updated, as to detail the new information.	As Expected.	N/A	

2.17[ii]) Teacher Record***2.17[ii][A]) Teacher Validation***

Test Number	Tested Data	Test Purpose	Additional Details	Expected Outcome	Actual Outcome	Corrective action	Page of Evidence
Category [ii]		Teacher Record					
Process: [A]:		Validation					
2.17[ii][B] [1]	First Name Validation	To ensure that the validation code is operational, and thus prevents or at least limits the instances for which users enter invalid characters.	{Textbox is blank}	An error message should appear, stating that the user must enter a value into the associated textbox.	As Expected.	N/A	
			{Numeric Character Entered}	An error message should appear, requesting that the user remove the numeric character.	As Expected.	N/A	
			{Symbolic Character Entered}	An error message should appear, requesting that the user remove the Symbol character.	As Expected.	N/A	

			{Punctual Character Entered}	An error message should appear, requesting that the user remove the punctuation character.	As Expected.	N/A	
			Text box max Length = 50;	The user should be restricted from entering more than 50 characters into this object.	As Expected.	N/A	
2.17[ii][B] [2]	Surname Validation	To ensure that the validation code is operational, and thus prevents or at least limits the instances for which users enter invalid characters.	{Textbox is blank}	An error message should appear, stating that the user must enter a value into the associated textbox.	As Expected.	N/A	
			{Numeric Character Entered}	An error message should appear, requesting that the user remove the numeric character.	As Expected.	N/A	

			{Symbolic Character Entered}	An error message should appear, requesting that the user remove the Symbol character.	As Expected.	N/A	
			{Punctual Character Entered}	An error message should appear, requesting that the user remove the punctuation character.	As Expected.	N/A	
			Text box max Length = 50;	The user should be restricted from entering more than 50 characters into this object.	As Expected.	N/A	
2.17[ii][B] [3]	Address Validation	To ensure that the validation code is operational, and thus prevents or at least limits the instances for	{Textbox is blank}	An error message should appear, stating that the user must enter a value into the associated textbox.	As Expected.	N/A	

		which users enter invalid characters.	{Numeric Character Entered}	An error message should appear, requesting that the user remove the numeric character.	As Expected.	N/A	
			{Symbolic Character Entered}	An error message should appear, requesting that the user remove the Symbol character.	As Expected.	N/A	
			{Punctual Character Entered}	An error message should appear, requesting that the user remove the punctuation character.	As Expected.	N/A	
			Text box max Length = 50;	The user should be restricted from entering more than 50 characters into this object.	As Expected.	N/A	

2.17[ii][B] [4]	Town Validation	To ensure that the validation code is operational, and thus prevents or at least limits the instances for which users enter invalid characters.	{Textbox is blank}	An error message should appear, stating that the user must enter a value into the associated textbox.	As Expected.	N/A	
			{Numeric Character Entered}	An error message should appear, requesting that the user remove the numeric character.	As Expected.	N/A	
			{Symbolic Character Entered}	An error message should appear, requesting that the user remove the Symbol character.	As Expected.	N/A	
			{Punctual Character Entered}	An error message should appear, requesting that the user remove the punctuation character.	As Expected.	N/A	

			Text box max Length = 50;	The user should be restricted from entering more than 50 characters into this object.	As Expected.	N/A	
2.17 [ii] [B] [5]	Post Code Validation	To ensure that the validation code is operational, and thus prevents or at least limits the instances for which users enter invalid characters.	{Textbox is blank}	An error message should appear, stating that the user must enter a value into the associated textbox.	As Expected.	N/A	
			[First two Characters = BT]	An error message should appear, stating that the first two characters must be "BT".	As Expected.	N/A	
			[Next Three Characters – Letters] {Numeric Character Present}	An error message should appear, stating that there is a numerical character present within the specified	As Expected.	N/A	

			character range, and must be removed.			
		[Next Three Characters – Letters] [Symbol Character Present]	An error message should appear, stating that there is a symbol character present within the specified character range, and must be removed.	As Expected.	N/A	
		[Next Three Characters – Letters] [Punctual Character Present]	An error message should appear, stating that there is a punctual character present within the specified character range, and must be removed.	As Expected.	N/A	
		[Final Three Characters – Numbers]	An error message should appear, stating that there	As Expected.	N/A	

			{Alphabetical Character Present}	is an Alphabetical character present within the specified character range, and must be removed.			
			[Final Three Characters – Numbers] {Symbol Character Present}	An error message should appear, stating that there is a symbol character present within the specified character range, and must be removed.	As Expected.	N/A	
			[Final Three Characters – Numbers] {Punctual Character Present}	An error message should appear, stating that there is a punctual character present within the specified character range, and must be removed.	As Expected.	N/A	

			{8 Characters Present}	Should the textbox contain less than 8 characters. An error message should appear, detain the format of the desired Post Code.	As Expected.	N/A	
2.17 [ii] [B] [6]	Email Address Validation	To ensure that the validation code is operational, and thus prevents or at least limits the instances for which users enter invalid characters.	{No Value Present}	An error message should appear, stating that this field must be completed.	As Expected.	N/A	
			{No "@" Present}	An error message should appear, stating that the Email Address value must contain a "@" symbol.	As Expected.	N/A	
			{Multiple "@"s Present}	An error message should appear, stating that an Email Address cannot contain multiple "@" symbols.	As Expected.	N/A	

			Text box max Length = 100;	The user should be restricted from entering more than 50 characters into this object.	As Expected.	N/A	
2.17 [ii] [B] [7]	Contact Number Validation	To ensure that the validation code is operational, and thus prevents or at least limits the instances for which users enter invalid characters.	{No value Present}	An error message should appear, stating that a value must be present.	As Expected.	N/A	
			{Letter Present}	An error message should appear, prompting the user to remove the alphabetical character.	As Expected.	N/A	
			{Symbol Present}	An error message should appear, prompting the user to remove the symbol character.	As Expected.	N/A	

			{Punctuation Present}	An error message should appear, prompting the user to remove the punctuation character.	As Expected.	N/A	
2.17[ii][B] [8]	Specialisation Validation	To ensure that the validation code is operational, and thus prevents or at least limits the instances for which users enter invalid characters.	{User Must Select Specialisation Value}	A combo box should ensure that the user must select a value. In addition to this, it should prevent the user from entering a custom value.	As Expected.	N/A	
2.17[ii][B] [9]	Room ID Validation	To ensure that the validation code is operational, and thus prevents or at least limits the instances for which users enter invalid characters.	{User Must Select Specialisation Value}	A combo box should ensure that the user must select a value. In addition to this, it should prevent the user from entering a custom value.	As Expected.	N/A	

2.17[ii][B]) Teacher Update Display

Test Number	Tested Data	Test Purpose	Additional Details	Expected Outcome	Actual Outcome	Corrective action	Page of Evidence
Category [ii]		Student Record					
Process [B]:		Validation					
2.17[ii][B] [1]	Teacher Update Display	To ensure that the correct information is displayed		Upon selecting the Teacher update button, the details associated with the selected Teacher record should appear within the form.	As Expected.	N/A	

2.17[iii]) Instrument Record***2.17[iii][A]) Instrument Record Validation***

Test Number	Tested Data	Test Purpose	Additional Details	Expected Outcome	Actual Outcome	Corrective action	Page of Evidence
Category [iii]		Instrument Record					
Process [A]:		Validation					
2.17[iii][C] [1]	Instrument Type Validation	To ensure that the validation code is operational, and thus prevents or at least limits the instances for which users enter invalid characters.	{User must select a value from the combo box} {Combo box will detail potential instrument groups: string, brass, woodwind, vocal, percussion and keyboard}	The user should be provided with the ability to easily select a value from the combo box. In addition to this, they should not possess the ability to enter a custom value into this field.	As Expected.	N/A	
2.17[iii][C] [2]	Instrument Name Validation	To ensure that the validation code is operational, and thus prevents or at least limits the instances for which users	{Textbox is blank}	An error message should appear, stating that the user must enter a value into the associated textbox.	As Expected.	N/A	

		enter invalid characters.	{Numeric Character Entered}	An error message should appear, requesting that the user remove the numeric character.	As Expected.	N/A	
			{Symbolic Character Entered}	An error message should appear, requesting that the user remove the Symbol character.	As Expected.	N/A	
			{Punctual Character Entered}	An error message should appear, requesting that the user remove the punctuation character.	As Expected.	N/A	
			Text box max Length = 50;	The user should be restricted from entering more than 50 characters into this object.	As Expected.	N/A	

2.17[iii][C] [3]	Quantity Validation	To ensure that the validation code is operational, and thus prevents or at least limits the instances for which users enter invalid characters.	{No value present}	In the event that the textbox is left blank, an error message should appear within the form, stating that a value must be entered.	As Expected.	N/A	
			{Presence of alphabetical character}	In the event that the entered value contains an alphabetical value, an error message should appear, prompting the user to remove the character.	As Expected.	N/A	
			{Presence of Symbol character}	In the event that the entered value contains a symbol value, an error message should appear, prompting the user to remove the character.	As Expected.	N/A	

			{Presence of Punctuation character}	In the event that the entered value contains a punctuation value, an error message should appear, prompting the user to remove the character.	As Expected.	N/A	
			{Entered value falls within the range of 0 and 19}	The numerical range which has been assigned to this text box is 1 – 19. This means that if the user enters a value which falls outside of this range, an error message should appear, stating the designated range, and prompt the user to modify the value.	As Expected.	N/A	

2.17[iii][B]) Instrument Record Update

Test Number	Tested Data	Test Purpose	Additional Details	Expected Outcome	Actual Outcome	Corrective action	Page of Evidence
Category [iii]		Student Record					
Process [B]:		Validation					
2.17[iii][B] [1]	Instrument Update Display	To ensure that the correct information is displayed		Upon selecting the Instrument update button, the details associated with the selected Instrument record should appear within the form.	As Expected.	N/A	

2.17[iv]) Room Record***2.17[iv][A]) Room Record Validation***

Test Number	Tested Data	Test Purpose	Additional Details	Expected Outcome	Actual Outcome	Corrective action	Page of Evidence
Category [iv]		Room Record					
Process [A]:		Validation					
2.17[iv][D] [1]	Room Type Validation	To ensure that the validation code is operational, and thus prevents or at least limits the instances for which users enter invalid characters.	{User must select a value from combo box} {Combo box should contain Room types: (Classroom, Hall, and Practice Room)}	The user should essentially be forced to select a value from the combo box. In addition to this, they should be restricted, as to prevent them from entering their own custom value.	As Expected.	N/A	
2.17[iv][D] [2]	Specialisation Validation	To ensure that the validation code is operational, and thus prevents or at least limits the instances for	{User must select a value from the combo box} {Combo box will detail potential instrument groups: string,}	The user should essentially be forced to select a value from the combo box. In addition to this, they should	As Expected.	N/A	

		which users enter invalid characters.	brass, woodwind, vocal, percussion and keyboard}	be restricted, as to prevent them from entering their own custom value.			
--	--	---------------------------------------	--	---	--	--	--

2.17[iv][B]) Room Record Update

Test Number	Tested Data	Test Purpose	Additional Details	Expected Outcome	Actual Outcome	Corrective action	Page of Evidence
Category [iv]		Room Record					
Process [B]:		Validation					
2.17[iv][B] [1]	Room Update Display	To ensure that the correct information is displayed		Upon selecting the Room update button, the details associated with the selected Room record should appear within the form.	As Expected.	N/A	

2.17[v]) Grade Record**2.17[v][A]) Grade Record Validation**

Test Number	Tested Data	Test Purpose	Additional Details	Expected Outcome	Actual Outcome	Corrective action	Page of Evidence
Category [v]		Grade Record					
Process [A]:		Validation					
2.17[v][E] [1]	Grade Title Validation	To ensure that the validation code is operational, and thus prevents or at least limits the instances for which users enter invalid characters.	{Textbox is blank}	An error message should appear, stating that the user must enter a value into the associated textbox.	As Expected.	N/A	
			{Numeric Character Entered}	An error message should appear, requesting that the user remove the numeric character.	As Expected.	N/A	
			{Symbolic Character Entered}	An error message should appear, requesting that the user remove the Symbol character.	As Expected.	N/A	

			{Punctual Character Entered}	An error message should appear, requesting that the user remove the punctuation character.	As Expected.	N/A	
			Text box max Length = 50;	The user should be restricted from entering more than 50 characters into this object.	As Expected.	N/A	
2.17 [v][E] [2]	Grade Fee Validation	To ensure that the validation code is operational, and thus prevents or at least limits the instances for which users enter invalid characters.	{No value present}	An error message should appear, stating that this textbox must possess a value.	As Expected.	N/A	
			{Alphabetical character present}	An error message should appear, stating that the invalid character has been identified, and then proceed to	As Expected.	N/A	

			<p>prompt the user to remove it.</p>			
		{Symbol character present}	<p>An error message should appear, stating that the invalid character has been identified, and then proceed to prompt the user to remove it.</p>	As Expected.	N/A	
		{Punctual character present}	<p>An error message should appear, stating that the invalid character has been identified, and then proceed to prompt the user to remove it.</p>	As Expected.	N/A	
		{Entered value must fall within selected range of: (1 – 100)}	<p>In the event that the user enters a value which falls outside of the designated range, an error message should appear,</p>	As Expected.	N/A	

				reiterating the assigned range, and prompting the user to enter a new value.			
--	--	--	--	--	--	--	--

2.17[v][B] Grade Record Update Display

Test Number	Tested Data	Test Purpose	Additional Details	Expected Outcome	Actual Outcome	Corrective action	Page of Evidence
Category [v]	Grade Record						
Process [B]:	Update						
2.17[v][B] [1]	Grade Update Display	To ensure that the correct information is displayed		Upon selecting the Grade update button, the details associated with the selected Grade record should appear within the form.	As Expected.	N/A	

2.17[vi]) Lesson Bundle Record***2.17[vi][A]) Lesson Bundle Record Validation***

Test Number	Tested Data	Test Purpose	Additional Details	Expected Outcome	Actual Outcome	Corrective action	Page of Evidence
Category [vi]		Lesson Bundle					
Process [A]:		Validation					
2.17[vi][F] [1]	Lesson Bundle Title Validation	To ensure that the validation code is operational, and thus prevents or at least limits the instances for which users enter invalid characters.	{Textbox is blank}	An error message should appear, stating that the user must enter a value into the associated textbox.	As Expected.	N/A	
			{Numeric Character Entered}	An error message should appear, requesting that the user remove the numeric character.	As Expected.	N/A	
			{Symbolic Character Entered}	An error message should appear, requesting that the user remove the Symbol character.	As Expected.	N/A	

			{Punctual Character Entered}	An error message should appear, requesting that the user remove the punctuation character.	As Expected.	N/A	
			Text box max Length = 50;	The user should be restricted from entering more than 50 characters into this object.	As Expected.	N/A	
2.17[vi][F] [2]	Bundle Cost Validation	To ensure that the validation code is operational, and thus prevents or at least limits the instances for which users enter invalid characters.	{No value present}	An error message should appear, stating that this textbox must possess a value.	As Expected.	N/A	
			{Alphabetical character present}	An error message should appear, stating that the invalid character has been identified, and	As Expected.	N/A	

				then proceed to prompt the user to remove it.			
			{Symbol character present}	An error message should appear, stating that the invalid character has been identified, and then proceed to prompt the user to remove it.	As Expected.	N/A	
			{Punctual character present}	An error message should appear, stating that the invalid character has been identified, and then proceed to prompt the user to remove it.	As Expected.	N/A	
			{Entered value must fall within selected range of: (1 – 60)}	In the event that the user enters a value which falls outside of the	As Expected.	N/A	

				designated range, an error message should appear, reiterating the assigned range, and prompting the user to enter a new value.			
2.17[vi][F] [3]	Multiplier Validation	To ensure that the validation code is operational, and thus prevents or at least limits the instances for which users enter invalid characters.	{No value present}	An error message should appear, stating that this textbox must possess a value.	As Expected.	N/A	
			{Alphabetical character present}	An error message should appear, stating that the invalid character has been identified, and then proceed to prompt the user to remove it.	As Expected.	N/A	

			{Symbol character present}	An error message should appear, stating that the invalid character has been identified, and then proceed to prompt the user to remove it.	As Expected.	N/A	
			{Punctual character present}	An error message should appear, stating that the invalid character has been identified, and then proceed to prompt the user to remove it.	As Expected.	N/A	
			{Entered value must fall within selected range of: (0 - 1)}	In the event that the user enters a value which falls outside of the designated range, an error message should appear, reiterating the assigned range,	As Expected.	N/A	

				and prompting the user to enter a new value.			
--	--	--	--	--	--	--	--

2.17[vi][B]) Lesson Bundle Record Update

Test Number	Tested Data	Test Purpose	Additional Details	Expected Outcome	Actual Outcome	Corrective action	Page of Evidence
Category [vi]		Lesson Bundle Record					
Process [B]:		Validation					
2.17[vi][B] [1]	Lesson Bundle Update Display	To ensure that the correct information is displayed		Upon selecting the Lesson Bundle update button, the details associated with the selected Lesson Bundle record should appear within the form.	As Expected.	N/A	

2.17[vii]) Purchased Lesson Record**2.17[vii][A]) Purchased Lesson Validation**

Test Number	Tested Data	Test Purpose	Additional Details	Expected Outcome	Actual Outcome	Corrective action	Page of Evidence
Category [vii]		Purchased Lessons					
Process [A]:		Validation					
2.17[vii][G] [1]	Student ID Validation	To ensure that the validation code is operational, and thus prevents or at least limits the instances for which users enter invalid characters.	{User must select a value from the combo box} {Combo box contain IDs and Names to represent all students currently present within the system}	The system should essentially force the user to select one of the values present within this combo box. In addition to this, it should restrict the user from entering their own custom value.	As Expected.	N/A	
2.17[vii][G] [2]	Lesson Bundle ID Validation	To ensure that the validation code is operational, and thus prevents or at least limits the	{User must select a value from the combo box} {Combo box contain IDs and Titles to represent}	The system should essentially force the user to select one of the values present within this combo box.	As Expected.	N/A	

		instances for which users enter invalid characters.	all Lesson Bundles currently present within the system}	In addition to this, it should restrict the user from entering their own custom value.			
2.17[vii][G] [3]	Purchase Date Validation	To ensure that the validation code is operational, and thus prevents or at least limits the instances for which users enter invalid characters.	{No value present}	In the event that there is no value present within the textbox, an error message should appear, stating that this object must contain a value.	As Expected.	N/A	
			{The user enters a value through the use of an associated calendar object}	Upon selecting a date value from the calendar, it should be added to the associated textbox. In addition to this, the user should be unable to directly add or modify	As Expected.	N/A	

				information within this object.			
2.17[vii][G] [4]	Payment Method Validation	To ensure that the validation code is operational, and thus prevents or at least limits the instances for which users enter invalid characters.	{User must select a value from the combo box} {Combo box will contain payment methods including: (Cash, Credit Card, Debit Card, Cheque, Other)}	The user should be essentially forced to select one of the values present within the associated combo box. In addition to this, they should be incapable of entering new or modifying data present within this component.	As Expected.	N/A	
2.17[vii][G] [5]	Payment Received Validation	To ensure that the validation code is operational, and thus prevents or at least limits the instances for which users enter invalid characters.	{Simple Check box to represent this value}	The user should possess the ability to check or un- check this object, as to dictate the state of the True/False Boolean.	As Expected.	N/A	

2.17[vii][G] [6]	Payment Received Date Validation	To ensure that the validation code is operational, and thus prevents or at least limits the instances for which users enter invalid characters.	{No value present}	In the event that no value is present within this object, an error message should appear, stating that this textbox must be completed.	As Expected.	N/A	
			{The user enters a value through the use of an associated calendar object}	Upon selecting a value present within the associated calendar object, the value should be added to the text box. This should ensure that it takes the correct format. In addition to this, the user should not possess the ability to directly modify or add information into this textbox.	As Expected.	N/A	

2.17[vii][G] [7]	Bundle Cost Validation	To ensure that the validation code is operational, and thus prevents or at least limits the instances for which users enter invalid characters.	{User will not directly modify this object}	<p>This field will be automatically calculated, as to ensure that the correct value is entered.</p> <p>It is important to note that in most cases, the user should not possess the ability to view or customize the contents of this object.</p>	As Expected.	N/A	
2.17[vii][G] [8]	Scheduled Validation	To ensure that the validation code is operational, and thus prevents or at least limits the instances for which users enter invalid characters.	{A simple Check box}	<p>This should provide the user with the ability to effectively dictate whether the condition is True or False, by modifying the state of the check box.</p>	As Expected.	N/A	

2.17[vii][B]) Purchased Lesson Record Update

Test Number	Tested Data	Test Purpose	Additional Details	Expected Outcome	Actual Outcome	Corrective action	Page of Evidence
Category [vii]		Purchased Lesson Record					
Process [B]:		Validation					
2.17[vii][B] [1]	Purchased Lesson Update Display	To ensure that the correct information is displayed		Upon selecting the Purchased Lesson update button, the details associated with the selected Purchased Lesson record should appear within the form.	As Expected.	N/A	

2.17[viii]) Scheduled Lesson Record**2.17[viii][A]) Scheduled Lesson Record Validation**

Test Number	Tested Data	Test Purpose	Additional Details	Expected Outcome	Actual Outcome	Corrective action	Page of Evidence
Category [viii]		Scheduled Lessons					
Process [A]:		Validation					
2.17[viii][H] [1]	Student ID Validation	To ensure that the validation code is operational, and thus prevents or at least limits the instances for which users enter invalid characters.	{User must select a value from the combo box} {Combo box will contain ID and Name values to represent all Students present within the system.}	The system should essentially force the user to select one of the values present within this combo box. In addition to this, it should restrict the user from entering their own custom value.	As Expected.	N/A	
2.17[viii][H] [2]	Teacher ID Validation	To ensure that the validation code is operational, and thus prevents or at least limits	{User must select a value from the combo box} {Combo box will contain ID and Name values to	The system should essentially force the user to select one of the values present within this combo box.	As Expected.	N/A	

		the instances for which users enter invalid characters.	represent all Teachers present within the system.}	In addition to this, it should restrict the user from entering their own custom value.			
2.17[viii][H] [3]	Purchased Lesson ID Validation	To ensure that the validation code is operational, and thus prevents or at least limits the instances for which users enter invalid characters.	{No Value Present}	In the event that there is no value present within this object, an error message should appear, stating that the user must enter a value.	As Expected.	N/A	
		{An associated button will cause a Data Grid to appear when selected} {This Grid will detail all purchased lessons}		Upon selecting one of the records, present within the Data Grid, the associated purchase ID should be presented within	As Expected.	N/A	

			associated with the selected student, as long as they have not been used}	the associated textbox.			
2.17[viii][H] [4]	Number of Week Validation	To ensure that the validation code is operational, and thus prevents or at least limits the instances for which users enter invalid characters.	{User must select a value from the combo box} {Combo box will contain information associated with the total weeks which could be selected: (5, 10, 15, 20 or 30 weeks)}	The user should essentially be forced to select one of the values present within this combo box. In addition to this, the user should also lack the ability to directly modify or add information in this object.	As Expected.	N/A	
2.17[viii][H] [5]	Start Date Validation	To ensure that the validation code is operational, and thus prevents or at least limits the instances for which	{No value present within textbox}	In the event that there is no value present within this textbox, an error message should appear, stating that the user must select a value.	As Expected.	N/A	

		users enter invalid characters.	{An associated button will cause a calendar to appear when selected}	Upon selecting a date value within the calendar object, the associated value should be presented within the associated textbox.	As Expected.	N/A	
2.17[viii][H] [6]	Booked Days Validation	To ensure that the validation code is operational, and thus prevents or at least limits the instances for which users enter invalid characters.	{No value present}	In the event that there is no value present within this field, an error message should appear, stating that the user must enter a value.	As Expected.	N/A	
			{User should not select more than 2 values}	Should the user select more than 2 values, an error message should appear, stating that they must select a maximum of 2.	As Expected.	N/A	

			<p>{An associated button should cause a custom menu to appear when selected}</p> <p>{This menu will display five check boxes, one for each of the days of the week}</p>	<p>Upon selecting the desired days from the custom menu, and closing it should cause the selected days information to transfer to the associated textbox.</p>	As Expected.	N/A	
2.17[viii][H] [7]	Booked Time Validation	To ensure that the validation code is operational, and thus prevents or at least limits the instances for which users enter invalid characters.	<p>{User must select a value from the combo box}</p> <p>{Combo box will contain information associated with the times which could be selected: (13:00 – 18:30)}</p>	<p>This combo box should essentially force the user to select one of the values store.</p> <p>In addition to this, it should also actively ensure that the user does not possess the ability to directly add or modify information in association to the</p>	As Expected.	N/A	

				combo box.			
2.17[viii][H] [8]	End Date Validation	To ensure that the validation code is operational, and thus prevents or at least limits the instances for which users enter invalid characters.	{No value present within textbox}	In the event that there is no value present within this textbox, and error message should appear, stating that the user must enter a value.	As Expected.	N/A	
			{An associated button will cause a calendar to appear when selected}	Upon selecting a date from the associated calendar, the value should be presented within the corresponding textbox. This effectively ensures that the user has no direct input of the information, and therefore validates the	As Expected.	N/A	

				format of the data.			
--	--	--	--	---------------------	--	--	--

2.17[viii][B]) Scheduled Lesson Record Update

Test Number	Tested Data	Test Purpose	Additional Details	Expected Outcome	Actual Outcome	Corrective action	Page of Evidence
Category [viii]		Scheduled Lesson Record					
Process [B]:		Update Display					
2.17[viii][B] [1]	Schedule Update Display	To ensure that the correct information is displayed		Upon selecting the Scheduled Lesson update button, the details associated with the selected Scheduled Lesson record should appear within the form.	As Expected.	N/A	

2.17[ix]) Additional Functions

Test Number	Tested Data	Test Purpose	Additional Details	Expected Outcome	Actual Outcome	Corrective action	Page of Evidence
Category [ix]		Additional Functions					
Process [A]:		Functions					
2.17[ix][A] [1]	Assign Lesson Dates from Schedule Records	To ensure that the desired records are being appended correctly.	This function is associated with the process of adding new schedule information into the system, be this entirely, or in the form of an update.	Upon selecting the add button, the system should automatically identify each lesson date for all scheduled lessons present within the system, and then proceed to store them within the Lesson Date Table.	As Expected.	N/A	
2.17[ix][A] [2]	Automatically calculate Bundle Cost	To ensure that the system possesses the ability to automatically calculate and update Bundle Costs.	This function is associated with the act of adding, or updating information for the purchased Lessons table.	Upon selecting the add / update button, the system should automatically acquire the necessary pieces of information to complete the calculation, this	As Expected.	N/A	

				including (Grade, Lesson Bundle and student), then use complete the calculation and finally update the associated record with the new information.			
2.17(ix)(A) [3]	Automatically calculate End Dates	To ensure that the system possesses the ability to automatically calculate and update the End Date.	This function is associated with the process of adding or updating information within the scheduled lesson table.	Upon selecting the add / update button, the system should automatically acquire information associated with the scheduled lesson, and then proceed to use the information to calculate an End Date value. Once calculated, it would then be updated into the associated record.	As Expected.	N/A	

2.18) Confirmation

frmConfirmation							
Test Number	Tested Data	Test Purpose	Additional Details	Expected Outcome	Actual Outcome	Corrective action	Page of Evidence
Process [A]: General							
2.3[A][1]	Selected Record Display	To ensure that the information is transferred correctly to this forms display.	The details that are being displayed within this form should be automatically updated to relate to the selected record, and their table of origin.	Upon accessing this form, the details associated with the selected record should appear within the display.	As Expected.	N/A	
2.3[A][2]	Confirm button	To ensure that this button completes the intended operation.		Upon selecting this button, the selected record should be permanently deleted from the system.	As Expected.	N/A	
2.3[A][3]	Return Button	To ensure that this button completes the intended operation.		Upon selecting this button ,the confirmation form should close, and return the user to the previous form.	As Expected.	N/A	

2.19) User Login

frmConfirmation							
Test Number	Tested Data	Test Purpose	Additional Details	Expected Outcome	Actual Outcome	Corrective action	Page of Evidence
Process [A]: General							
2.3[A][1]	User Login (Correct Details)	To ensure that this button completes the intended operation.	N/A	Should the user have entered the correct login details, upon selecting button, the form should close, and the user should now be logged in.	As Expected.	N/A	
2.3[A][2]	User Login (Incorrect Details)	To ensure that this button completes the intended operation.	N/A	Should the user select this button with a incorrect values present within the textboxes, an error message should appear.	As Expected.	N/A	
2.3[A][3]	Return Button	To ensure that this button completes the intended operation.	N/A	Upon selecting this button ,the confirmation form should close, and return the user to the previous form.	As Expected.	N/A	

2.20) Invoice Form

frm							
Test Number	Tested Data	Test Purpose	Additional Details	Expected Outcome	Actual Outcome	Corrective action	Page of Evidence
Process [A]:		Display					
2.20[A][1]	Purchase Record Information	To ensure that the correct information is displayed within the form.	This information will be expressed through the use of a collection of lists, and custom displays.	Upon accessing this system should identify all purchased lessons associated with the selected student that have not been paid, and then proceed to display them.	As Expected	N/A	
2.20[A][2]	Total Bundle Cost	To ensure that the correct information is displayed within the form.	This is a simple operation that will display a total of all bundle cost label values.	Upon accessing this form, the system should then review all bundle cost labels present, and then complete a simple calculation to acquire a grand total, which will be	As Expected	N/A	

				displayed within a label on the form.			
2.20[A][3]	Selected Student Name	To ensure that the correct information is displayed within the form.	N/A	The student name label present within this form should update to reflect the selected student for this form.	As Expected	N/A	
2.20[A][4]	Error Message	To ensure that the correct information is displayed within the form.	N/A	In the event that the selected student does not possess any outstanding lesson payments, then this error message should appear with this statement.	As Expected	N/A	
Process [B]:		Form Navigation					
2.20[B][1]	Return to Main Menu	To ensure that the intended operation is	N/A	Selecting this button should cause this form to hide, and the	As Expected	N/A	

		completed upon selecting the associated button.		private tuition form menu to appear.			
		Print Operation					
2.20[C][1]	Print button	To ensure that the intended operation is completed upon selecting the associated button.	N/A	Selecting this button should cause a print preview of this form to appear.	The print preview display will appear, however it does not seem that the page would fit the entirety of this form.	This was a simple proof of concept, and in theory is operational, therefore it is a problem which will be addressed further down through the development process.	

Test Evidence

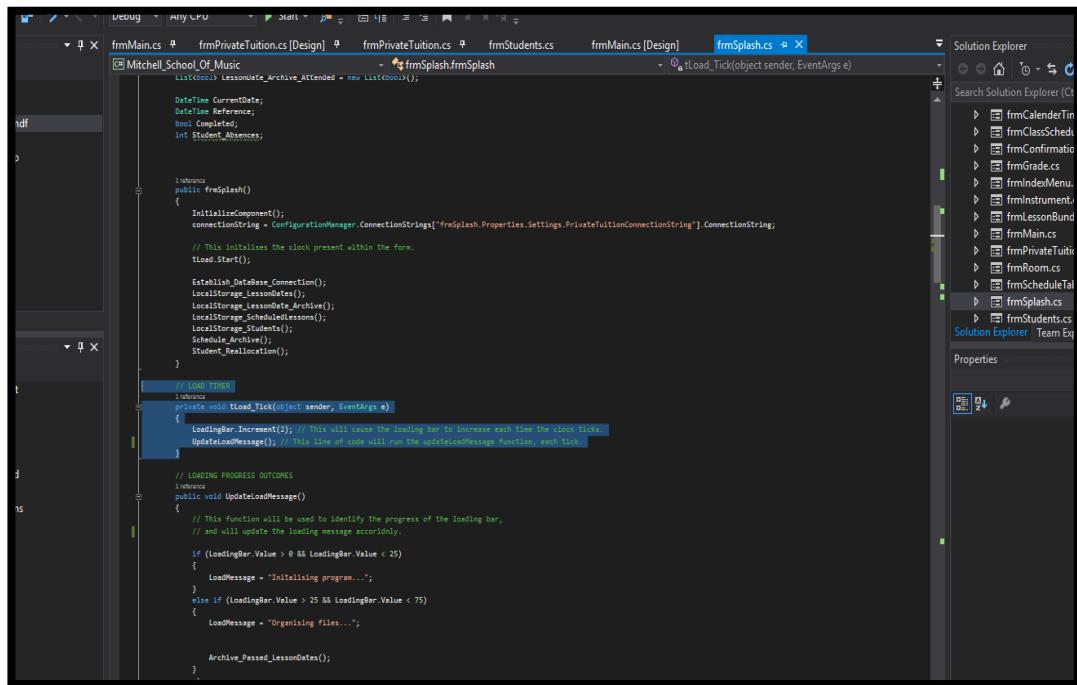
2) Forms

2.1) Splash Screen

Process [A]) Loading Bar Progress + Feedback

2.1[A][1] – Loading Bar Value

2.1[A][1] Screenshot A



```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Mitchell_School_Of_Music;
using System.Data.SqlClient;
using System.Configuration;
using System.IO;
using System.Threading;

namespace Mitchell_School_Of_Music
{
    public partial class frmSplash : Form
    {
        List<Lesson> LessonArchive_Attended = new List<Lesson>();
        DateTime CurrentDate;
        DateTime Reference;
        bool Completed;
        int Student_Absences;

        public frmSplash()
        {
            InitializeComponent();
            connectionString = ConfigurationManager.ConnectionStrings["frmSplash.Properties.Settings.PrivateTuitionConnectionString"].ConnectionString;
            // This initialises the clock present within the form.
            timer1.Start();

            Establish_Database_Connection();
            localStorage_LessonData();
            localStorage_LessonDate_Archive();
            localStorage_ScheduledLessons();
            localStorage_Students();
            Schedule_Archive();
            Student_Reallocation();
        }

        #region TIME
        //Indicates
        private void timer1_Tick(object sender, EventArgs e)
        {
            loadingBar.Increment(2); // This will cause the loading bar to increase each time the clock ticks.
            UpdateLoadMessage(); // This line of code will run the UpdateLoadMessage function, when ticks.
        }
        #endregion TIME

        // LOADING PROGRESS OUTCOMES
        //Indicates
        public void UpdateLoadMessage()
        {
            // This function will be used to identify the progress of the loading bar,
            // and will update the loading message accordingly.

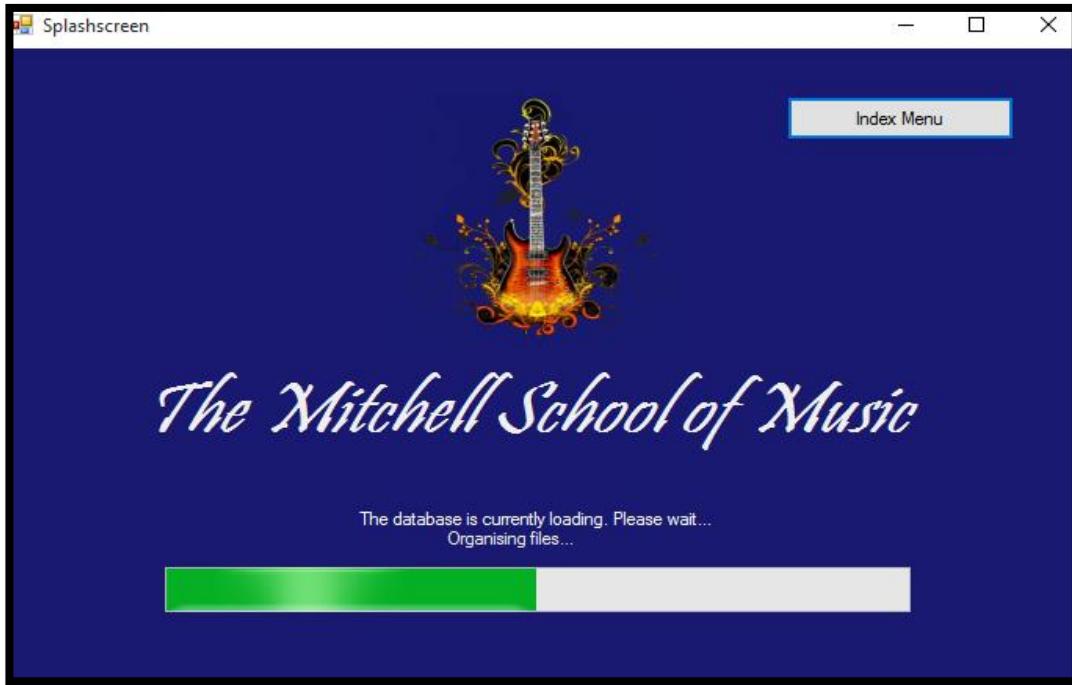
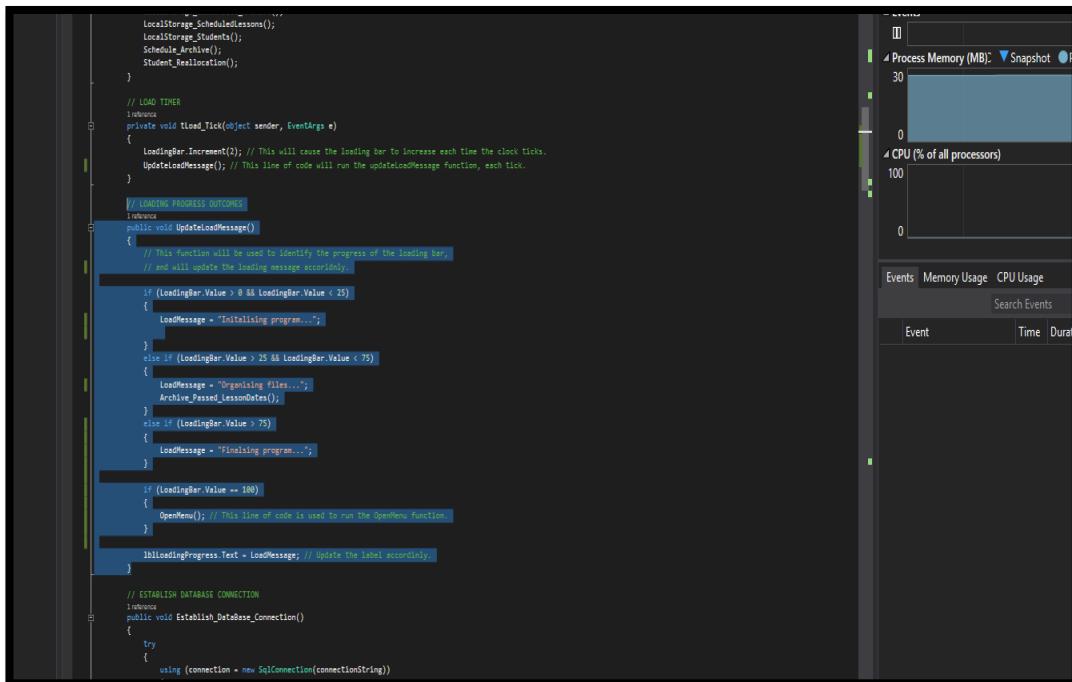
            if (loadingBar.Value > 0 && loadingBar.Value < 25)
            {
                loadMessage = "Initialising program...";
            }
            else if ((loadingBar.Value > 25 && loadingBar.Value < 75)
            {
                loadMessage = "Organising files...";
            }

            Archive_Passed_LessonNotes();
        }
    }
}

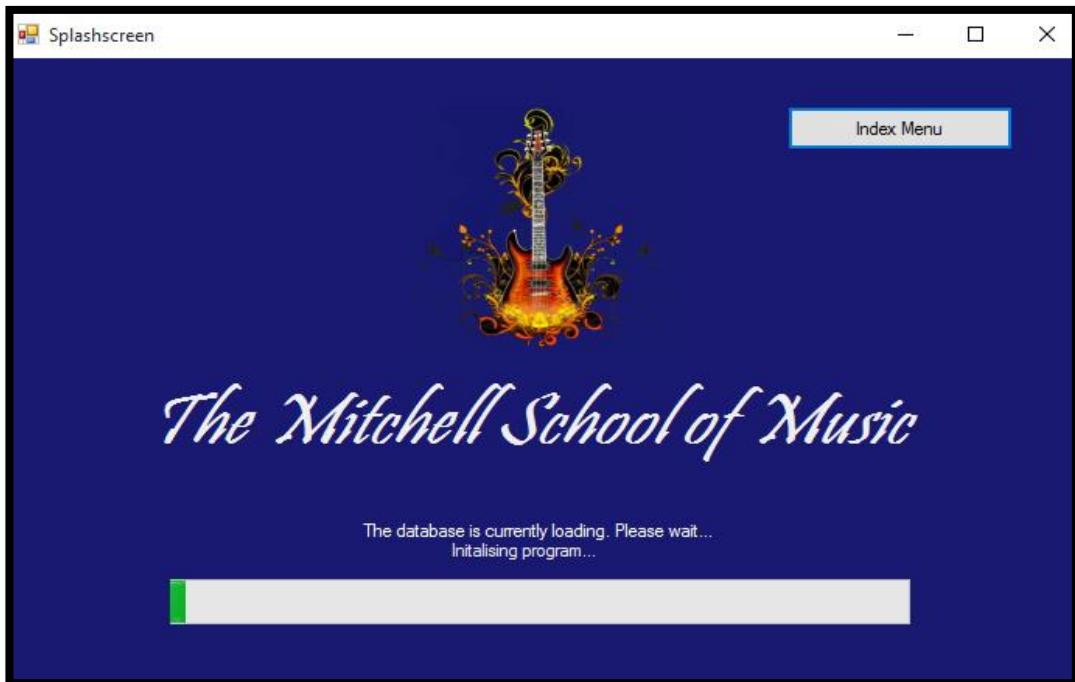
```

2.1[A][1] Screenshot B

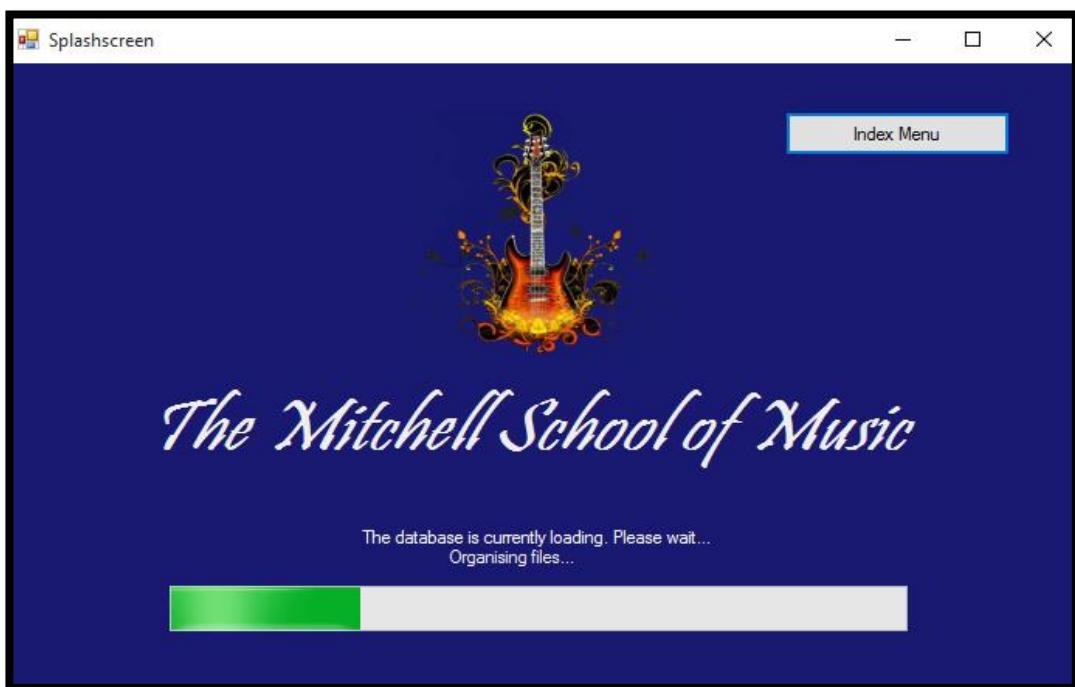


2.1[A][1] Screenshot C**2.1[A][2] – Load Message Update****2.1[A][2] Screenshot A**

2.1[A][2] Screenshot B



2.1[A][2] Screenshot C

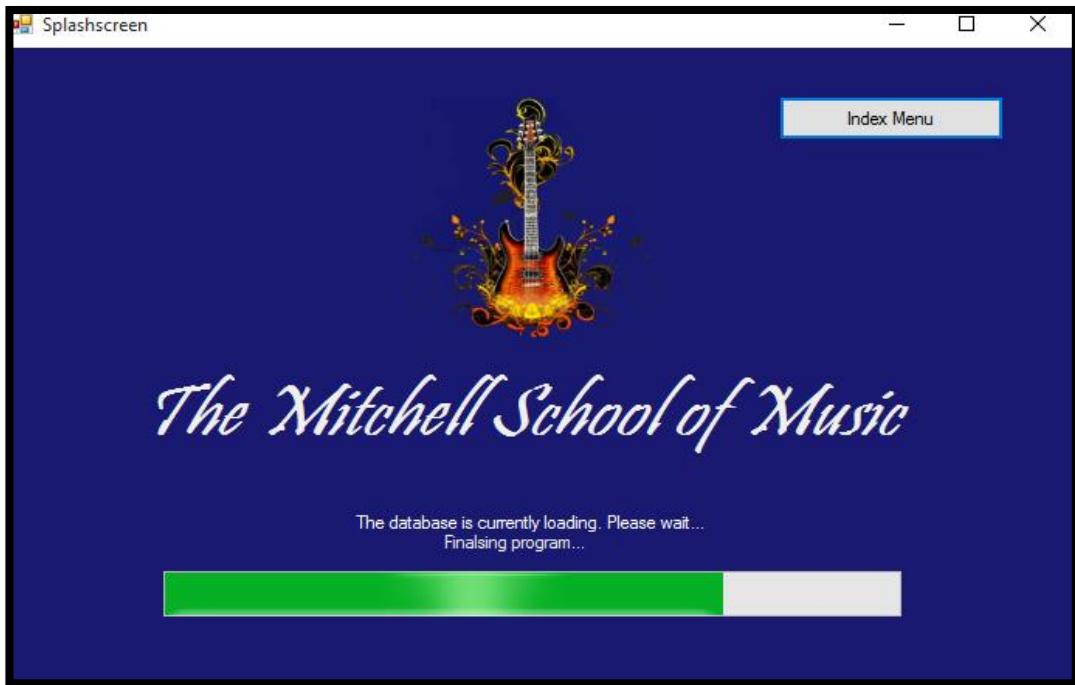


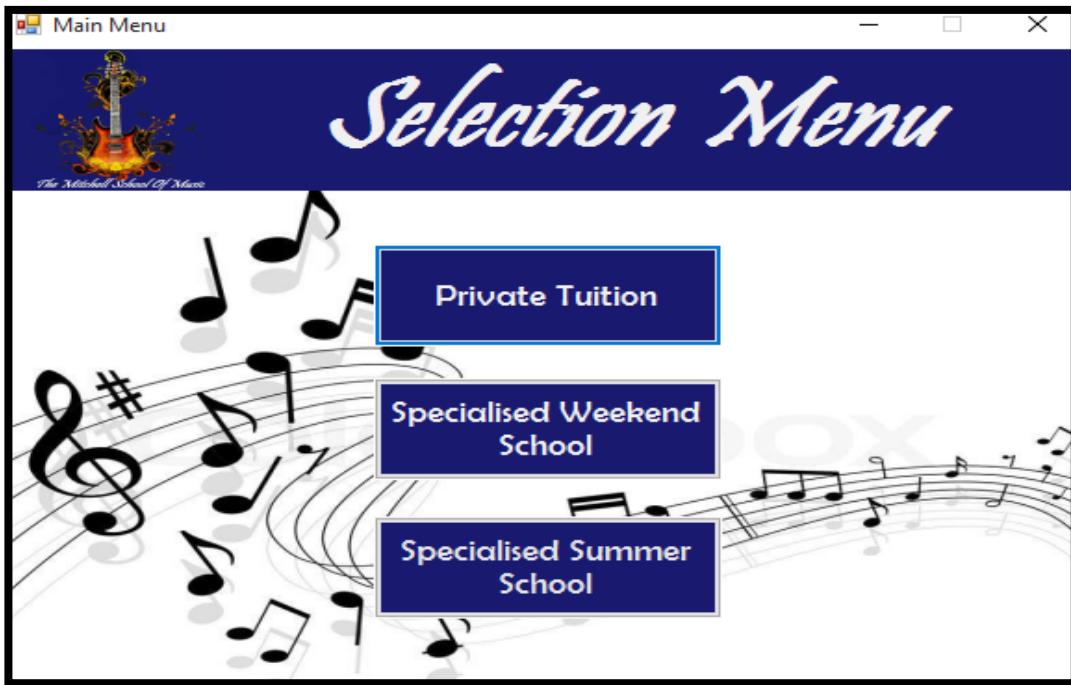
2.1[A][2] Screenshot D



2.1[A][3] – Load Main Form

2.1[A][3] Screenshot A



2.1[A][3] Screenshot B**Process [B]) Database Information Access + storage****2.1[B][1] – Student Storage List****2.1[B][1] Screenshot A**

```
// STORE STUDENT INFORMATION
-inherits
public void LocalStorage_Students()
{
    using (connection = new SqlConnection(connectionString))
    {
        connection.Open();
        using (SqlCommand cmd = new SqlCommand("SELECT * FROM Students", connection))
        {

            DataReader = cmd.ExecuteReader();
            while (DataReader.Read())
            {
                Students_ID.Add(DataReader.GetInt32(0));
                Students_FirstName.Add(DataReader.GetString(1));
                Students_OtherNames.Add(DataReader.GetString(2));
                Students_Surname.Add(DataReader.GetString(3));
                Students_DateOfBirth.Add(DataReader.GetDateTime(4));
                Students_Address.Add(DataReader.GetString(5));
                Students_Town.Add(DataReader.GetString(6));
                Students_PostCode.Add(DataReader.GetString(7));
                Students_ContactNumber.Add(DataReader.GetInt32(8));
                Students_EmailAddress.Add(DataReader.GetString(9));
                Students_GradeID.Add(DataReader.GetInt32(10));
                Students_InstrumentID.Add(DataReader.GetInt32(11));
                Students_Payment.Add(DataReader.GetBoolean(12));
                Reallocation.StudentAbsences.Add(0);

                MessageBox.Show("Student Information: " + Students_ID[test] + " " + Students_FirstName[test] + " " + Students_OtherNames[test] + " " + Students_Surname[test] + " " + Students_DateOfBirth[test]
                    + " " + Students_Address[test] + " " + Students_Town[test] + " " + Students_PostCode[test] + " " + Students_ContactNumber[test] + " " + Students_EmailAddress[test] + " " + Students_GradeID[test]
                    + " " + Students_InstrumentID[test] + " " + Students_Payment[test]);
                test++;
            }
        }
    }

    // STORE SCHEDULED LESSONS INFORMATION
    -inherits
    public void LocalStorage_ScheduledLessons()
    {
        using (connection = new SqlConnection(connectionString))
        {
            connection.Open();
            using (SqlCommand cmd = new SqlCommand("SELECT * FROM Scheduled Lessons", connection))
            {

            }
        }
    }
}
```

2.1[B][1] Screenshot B

The screenshot shows a Microsoft Access application window with multiple tabs at the top: frmMain.cs, frmPrivateTuition.cs [Design], frmPrivateTuition.cs, dbo.Students [Data] (highlighted), frmStudents.cs, frmMain.cs [Design], frmSplash.cs, and frmSplash.cs [Design]. Below the tabs is a grid view of a table with 27 rows of student data. A modal dialog box is overlaid on the grid, containing the following text:

Student Information: 4 Linda Louise Simmons 20/02/1998 00:00:00 14 Greenroad Coleraine BT51 2HE 70359371 Linda854@gmail.com 2 True

OK

StudentID	First_Name	Other_Name(s)	Surname	DateOfBirth	Address	Town	PostCode	Contact_Num..	Email_Address	GradeID	InstrumentID	Tuition_Fee_R..
4	Linda	Louise	Simmons	20/02/1998	14 Greenroad	Coleraine	BT51 2HE	70359371	Linda854@gmail.com	2	1	True
5	Trevor	Luke	Simmons	20/08/1997	22 The Glebe	Ballymoney	BT32 3ND	74920033	Trevorsimmons..	1	4	True
6	Daniel	Robert	Waters	01/02/1970	199 Pacroad	Ballymoney	BT47 3MF	37719048	danielwaters99..	4	3	True
7	Adam	Brent	Ackles	30/06/1993	731 boathane	Garvagh	BT43 2HT	46299567	adambrent@ya..	2	2	False
8	Jenson	James	Tweed	22/12/1990	71 glebe avenue	Ballycastle	BT37 5GW	33923231	jttweed@hotmail..	4	5	True
9	Diana	Ann	James	14/04/1985	6 Polyroad	Bushmills	BT83 5FE	40059237	dianajames@ya..	4	6	True
10	Lorraine	Hannah	Silvers	17/08/1999	48 union street	Portrush	BT84 2TR	40027364	loraine47silvers..	1	7	False
11	Courtney	Lisa	Doherty	13/01/1994	11 jacobs lane	Garvagh	BT95 3RE	47326678	courtney00@ya..	1	8	True
12	Patrick	Samuel	Roberts	20/02/1973	92 Ballyway	Ballybogey	BT05 5RF	30283736	patrick26@yahoo..	2	9	True
13	Matthew	Nathan	Jones	26/06/1975	45 tinkers street	Portstewart	BT82 7EQ	46335362	matthewjones..	3	2	False
14	Samantha	Alison	Mc farlane	17/05/1982	231 Anderson A..	Ballymoney	BT00 3UN	45673623	sammym4@ya..	2	4	True
15	Linda	Hayley	Spears	04/04/1996	999 Goldhill	Ballymena	BT71 5YR	37434794	lindahspears29..	3	5	False
16	Lucy	Susan	Dysart	28/07/1991	52 Colour avenue	Coleraine	BT52 1QA	50382627	tonyhoward@g..	4	6	True
17	Anthony	Ryan	Howard	12/03/1990	32 Hillview	Portrush	BT95 3YU	50947464	tonyhoward@q..	3	10	True
18	Cleo	Laura	Kane	23/02/2000	32 Hillview	Ballybogey	BT59 2EC	3892374	cleopatra6@ya..	4	2	True
19	Louise	Amanda	Doran	04/09/1994	39 Union street	Coleraine	BT51 3EH	75434934	louloudoran009..	3	5	True
20	Gerard	Josh	Mc daid	25/04/1968	23 Chief Street	Bushmills	BT78 4YT	46637328	gerard300@gm..	1	7	False
21	Kayla	Brooke	Black	13/10/1978	91 butchroad	Ballymoney	BT94 2YU	53752852	kaylablack@ya..	2	6	False
22	Kristeen	Casey	Dickenson	14/11/1979	19 Screenhill ro..	Ballycastle	BT39 5GF	53278523	Kdickenson@ya..	2	10	False
23	Leanna	Skylar	Philips	01/01/1992	124 Strand Park	Garvagh	BT43 9KV	96854746	LeaSky942@Ho..	1	3	False
24	Duke	Kassy	Ewart	25/12/1998	220 Anderson A..	Portrush	BT59 4RM	57325632	Dukeewart@G..	1	3	False
25	Alan	Shantelle	Blue	21/08/2995	60 Tullyarton D..	BallyBogey	BT99 0CJ	53275823	AlanBlue@Yah..	4	2	True
26	Reginald	Jessa	Stevenson	05/07/1982	14 Gelbe Park	Coleraine	BT46 2HD	13123534	rjs24@gmail.co..	1	9	False
27	Frederica	Trevelyan	Cobb	09/03/2000	84 Crescent Road	Coleraine	BT59 5GL	70329356	FredCobb@Hot..	3	4	True
o	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

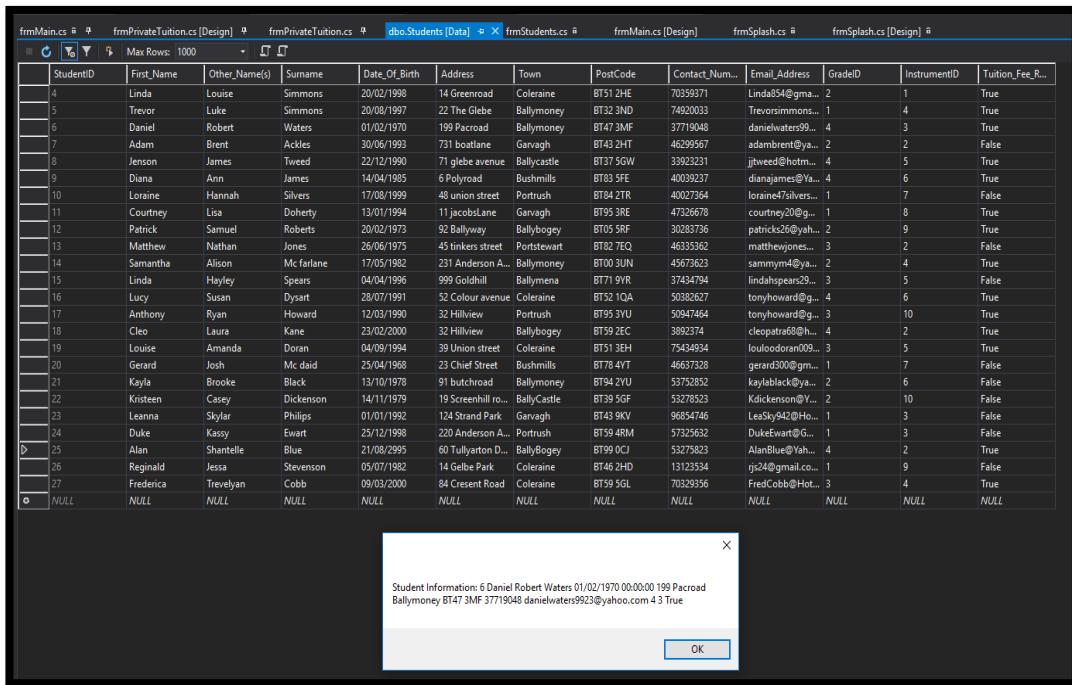
2.1[B][1] Screenshot C

The screenshot shows a Microsoft Access application window with multiple tabs at the top: frmMain.cs, frmPrivateTuition.cs [Design], frmPrivateTuition.cs, dbo.Students [Data] (highlighted), frmStudents.cs, frmMain.cs [Design], frmSplash.cs, and frmSplash.cs [Design]. Below the tabs is a grid view of a table with 27 rows of student data. A modal dialog box is overlaid on the grid, containing the following text:

Student Information: 5 Trevor Luke Simmons 20/08/1997 00:00:00 22 The Glebe Ballymoney BT32 3ND 74920033 Trevorsimmons47@gmail.com 14 True

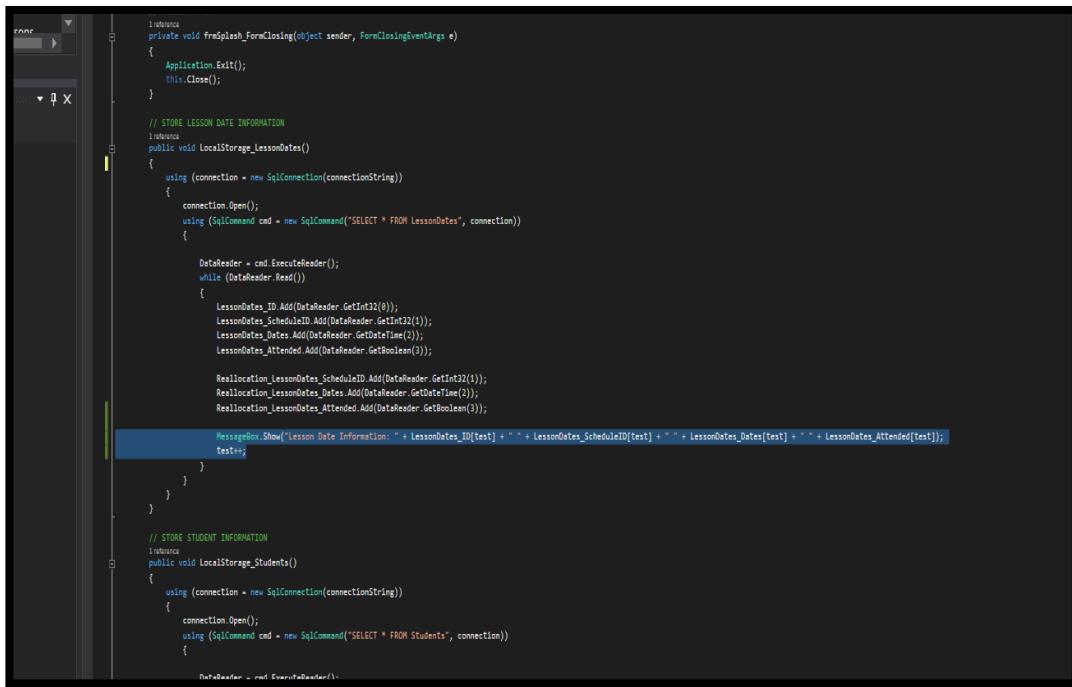
OK

StudentID	First_Name	Other_Name(s)	Surname	DateOfBirth	Address	Town	PostCode	Contact_Num..	Email_Address	GradeID	InstrumentID	Tuition_Fee_R..
4	Linda	Louise	Simmons	20/02/1998	14 Greenroad	Coleraine	BT51 2HE	70359371	Linda854@gmail.com	2	1	True
5	Trevor	Luke	Simmons	20/08/1997	22 The Glebe	Ballymoney	BT32 3ND	74920033	Trevorsimmons..	1	4	True
6	Daniel	Robert	Waters	01/02/1970	199 Pacroad	Ballymoney	BT47 3MF	37719048	danielwaters99..	4	3	True
7	Adam	Brent	Ackles	30/06/1993	731 boathane	Garvagh	BT43 2HT	46299567	adambrent@ya..	2	2	False
8	Jenson	James	Tweed	22/12/1990	71 glebe avenue	Ballycastle	BT37 5GW	33923231	jttweed@hotmail..	4	5	True
9	Diana	Ann	James	14/04/1985	6 Polyroad	Bushmills	BT83 5FE	40059237	dianajames@ya..	4	6	True
10	Lorraine	Hannah	Silvers	17/08/1999	48 union street	Portrush	BT84 2TR	40027364	loraine47silvers..	1	7	False
11	Courtney	Lisa	Doherty	13/01/1994	11 jacobs lane	Garvagh	BT95 3RE	47326678	courtney00@ya..	1	8	True
12	Patrick	Samuel	Roberts	20/02/1973	92 Ballyway	Ballybogey	BT05 5RF	30283736	patrick26@yahoo..	2	9	True
13	Matthew	Nathan	Jones	26/06/1975	45 tinkers street	Portstewart	BT82 7EQ	46335362	matthewjones..	3	2	False
14	Samantha	Alison	Mc farlane	17/05/1982	231 Anderson A..	Ballymoney	BT00 3UN	45673623	sammym4@ya..	2	4	True
15	Linda	Hayley	Spears	04/04/1996	999 Goldhill	Ballymena	BT71 5YR	37434794	lindahspears29..	3	5	False
16	Lucy	Susan	Dysart	28/07/1991	52 Colour avenue	Coleraine	BT52 1QA	50382627	tonyhoward@g..	4	6	True
17	Anthony	Ryan	Howard	12/03/1990	32 Hillview	Portrush	BT95 3YU	50947464	tonyhoward@q..	3	10	True
18	Cleo	Laura	Kane	23/02/2000	32 Hillview	Ballybogey	BT59 2EC	3892374	cleopatra6@ya..	4	2	True
19	Louise	Amanda	Doran	04/09/1994	39 Union street	Coleraine	BT51 3EH	75434934	louloudoran009..	3	5	True
20	Gerard	Josh	Mc daid	25/04/1968	23 Chief Street	Bushmills	BT78 4YT	46637328	gerard300@gm..	1	7	False
21	Kayla	Brooke	Black	13/10/1978	91 butchroad	Ballymoney	BT94 2YU	53752852	kaylablack@ya..	2	6	False
22	Kristeen	Casey	Dickenson	14/11/1979	19 Screenhill ro..	Ballycastle	BT39 5GF	53278523	Kdickenson@ya..	2	10	False
23	Leanna	Skylar	Philips	01/01/1992	124 Strand Park	Garvagh	BT43 9KV	96854746	LeaSky942@Ho..	1	3	False
24	Duke	Kassy	Ewart	25/12/1998	220 Anderson A..	Portrush	BT59 4RM	57325632	Dukeewart@G..	1	3	False
25	Alan	Shantelle	Blue	21/08/2995	60 Tullyarton D..	BallyBogey	BT99 0CJ	53275823	AlanBlue@Yah..	4	2	True
26	Reginald	Jessa	Stevenson	05/07/1982	14 Gelbe Park	Coleraine	BT46 2HD	13123534	rjs24@gmail.co..	1	9	False
27	Frederica	Trevelyan	Cobb	09/03/2000	84 Crescent Road	Coleraine	BT59 5GL	70329356	FredCobb@Hot..	3	4	True
o	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

2.1[B][1] Screenshot D


The screenshot shows a Windows application window with multiple tabs at the top: frmMain.cs, frmPrivateTuition.cs [Design], frmPrivateTuition.cs, dbo.Students [Data] (highlighted), frmStudents.cs, frmMain.cs [Design], frmSplash.cs, and frmSplash.cs [Design]. Below the tabs is a DataGridView containing student data. A modal dialog box is displayed in the center, titled "Student Information". The dialog contains the text: "Student Information: 6 Daniel Robert Waters 01/02/1970 00:00:00 199 Pacroad Ballymoney BT47 3MF 37719048 danielwaters9923@yahoo.com 4 3 True". At the bottom of the dialog is an "OK" button.

StudentID	First_Name	Other_Name(s)	Surname	DateOfBirth	Address	Town	PostCode	Contact_Num..	Email_Address	GradelD	InstrumentID	Tuition_Fee_R...
4	Linda	Louise	Simmons	20/02/1998	14 Greenroad	Coleraine	BT51 2HE	70359371	Linda54@gmail...	2	1	True
5	Trevor	Luke	Simmons	20/08/1997	22 The Glebe	Ballymoney	BT32 2ND	7490033	Trevorsimmons...	1	4	True
6	Daniel	Robert	Waters	01/02/1970	199 Pacroad	Ballymoney	BT47 3MF	37719048	danielwaters99...	4	3	True
7	Adam	Brent	Ackles	30/06/1993	731 boatlane	Garvagh	BT43 2HT	4629567	adambrett@ya...	2	2	False
8	Jenson	James	Tweed	22/12/1990	71 plebe avenue	Ballycastle	BT37 5GW	33923231	jtwedd@hotmail...	4	5	True
9	Diana	Anne	James	14/04/1985	6 Polyroad	Bushmills	BT83 5FE	40039237	dianajames@Ya...	4	6	True
10	Lorraine	Hannah	Silvers	17/08/1999	48 union street	Portrush	BT84 2TR	40027364	lorain47silvers...	1	7	False
11	Courtney	Lisa	Doherty	13/01/1994	11 jacobsLane	Garvagh	BT95 3RE	47326678	courtney200g...	1	8	True
12	Patrick	Samuel	Roberts	20/02/1973	92 Ballyway	Ballybogey	BT05 5RF	30283736	patrick23@yah...	2	9	True
13	Matthew	Nathan	Jones	26/06/1975	45 tinkers street	Portstewart	BT82 TEQ	46335362	matthewjones...	3	2	False
14	Samantha	Alison	Mc farlane	17/05/1982	231 Anderson A...	Ballymoney	BT00 3UN	45673623	sammym4@ya...	2	4	True
15	Linda	Hayley	Spears	04/04/1996	999 Goldhill	Ballymena	BT71 5YR	37434794	lindahpears29...	3	5	False
16	Lucy	Susan	Dysart	28/07/1991	52 Colour avenue	Coleraine	BT52 1QA	50382627	tonyhoward@q...	4	6	True
17	Anthony	Ryan	Howard	12/03/1990	32 Hillview	Portrush	BT95 3YU	50947464	tonyhoward@q...	3	10	True
18	Cleo	Laura	Kane	23/02/2000	32 Hillview	Ballybogey	BT59 2EC	3892374	leepatrat68@h...	4	2	True
19	Louise	Amanda	Doran	04/09/1994	39 Union street	Coleraine	BT51 2EH	75454934	louloedoran09...	3	5	True
20	Gerard	Josh	Mc daid	25/04/1968	23 Chief Street	Bushmills	BT78 4YT	46637328	gerard300@gm...	1	7	False
21	Kayla	Brooke	Black	13/10/1978	91 butchroad	Ballymoney	BT94 2VU	53752852	kaylablack@ya...	2	6	False
22	Kristeen	Casey	Dickenson	14/11/1979	19 Screenhill ro...	BallyCastle	BT39 5GF	53278523	Kdickenson@Y...	2	10	False
23	Leanna	Skylar	Philips	01/01/1992	124 Strand Park	Garvagh	BT43 3KV	96854746	LeaSky942@Ho...	1	3	False
24	Duke	Kassy	Ewart	25/12/1998	220 Anderson A...	Portrush	BT59 4RM	57325632	DukeEwart@G...	1	3	False
25	Alan	Shantelle	Blue	21/08/2995	60 Tullyarton D...	BallyBogey	BT99 0CJ	53275823	AlanBlue@Yah...	4	2	True
26	Reginald	Jessa	Stevenson	05/07/1982	14 Gelbe Park	Coleraine	BT46 2HD	13123534	rjs24@gmail.co...	1	9	False
27	Frederica	Trevlyan	Cobb	09/03/2000	84 Crescent Road	Coleraine	BT59 5GL	70329356	FredCobb@Hot...	3	4	True
o	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

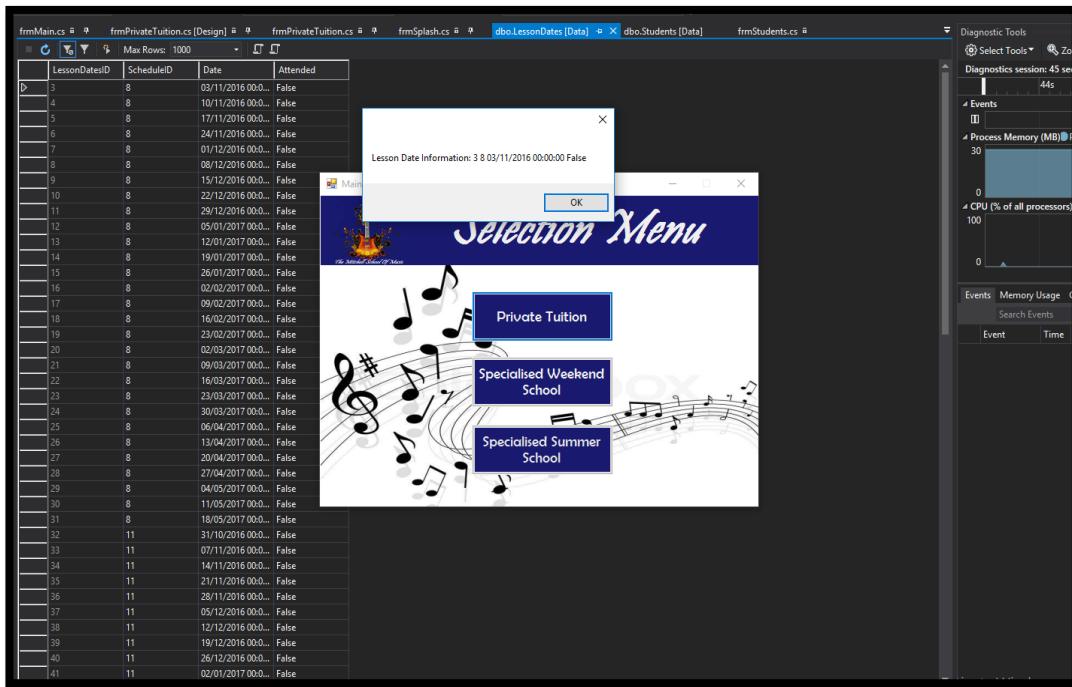
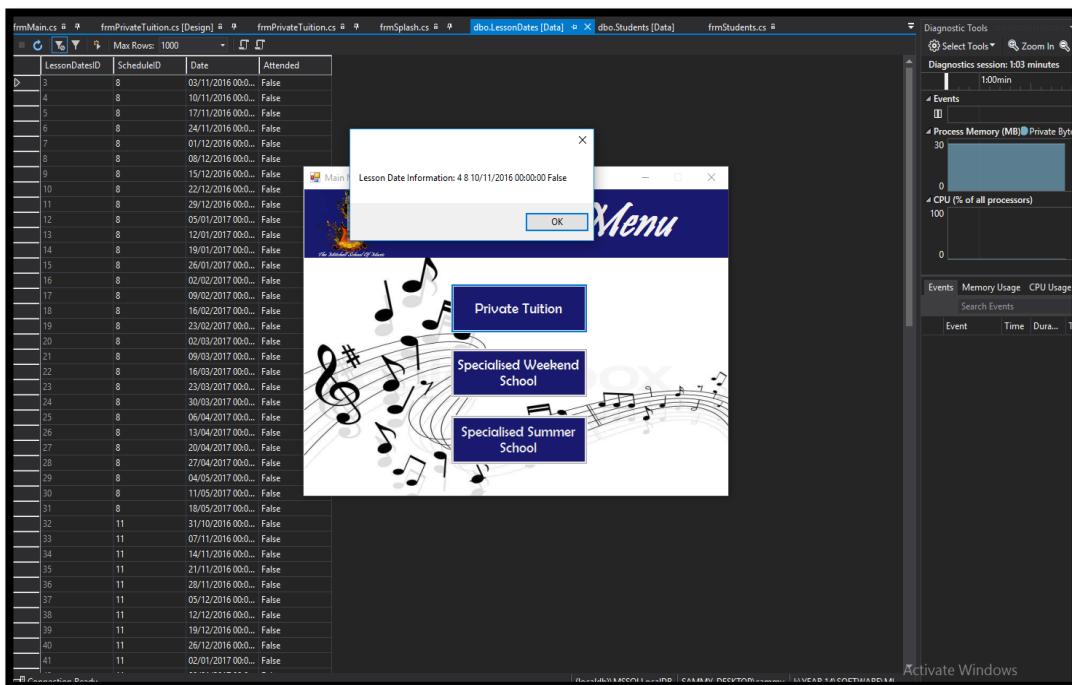
2.1[B][2] – Lesson Date Storage List**2.1[B][2] Screenshot A**


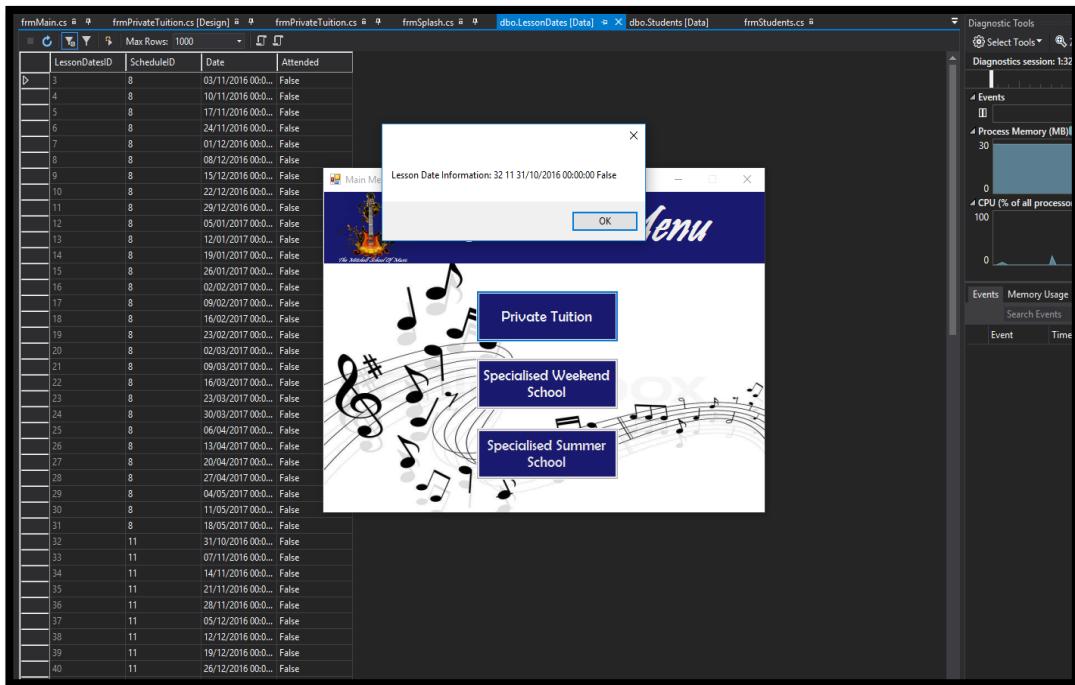
The screenshot shows the Visual Studio code editor with C# code for a Windows application. The code includes methods for storing lesson dates and student information. The `frmSplash` class has a `FormClosing` event handler that exits the application. The `LocalStorage_LessonDates` method uses a SQL connection to insert data into the `LessonDates` table. The `LocalStorage_Students` method also uses a SQL connection to insert data into the `Students` table. A message box is shown in the code, displaying a string that includes lesson dates and student details.

```

    1 //REFERENCES
    2 private void frmSplash_FormClosing(object sender, FormClosingEventArgs e)
    3 {
    4     Application.Exit();
    5     this.Close();
    6 }
    7
    8 // STORE LESSON DATE INFORMATION
    9 public void LocalStorage_LessonDates()
   10 {
   11     using (connection = new SqlConnection(connectionString))
   12     {
   13         connection.Open();
   14         using (SqlCommand cmd = new SqlCommand("SELECT * FROM LessonDates", connection))
   15         {
   16             DataReader = cmd.ExecuteReader();
   17             while (DataReader.Read())
   18             {
   19                 LessonDates_ID.Add(DataReader.GetInt32(0));
   20                 LessonDates_ScheduleID.Add(DataReader.GetInt32(1));
   21                 LessonDates_Dates.Add(DataReader.GetDateTime(2));
   22                 LessonDates_Attended.Add(DataReader.GetBoolean(3));
   23
   24                 ResAllocation_LessonDates_ScheduleID.Add(DataReader.GetInt32(1));
   25                 ResAllocation_LessonDates_Dates.Add(DataReader.GetDateTime(2));
   26                 ResAllocation_LessonDates_Attended.Add(DataReader.GetBoolean(3));
   27
   28                 MessageBox.Show("Lesson Date Information: " + LessonDates_ID[test] + " " + LessonDates_ScheduleID[test] + " " + LessonDates_Dates[test] + " " + LessonDates_Attended[test]);
   29             }
   30         }
   31     }
   32
   33 // STORE STUDENT INFORMATION
   34 public void LocalStorage_Students()
   35 {
   36     using (connection = new SqlConnection(connectionString))
   37     {
   38         connection.Open();
   39         using (SqlCommand cmd = new SqlCommand("SELECT * FROM Students", connection))
   40         {
   41             DataReader = cmd.ExecuteReader();
   42         }
   43     }
   44 }

```

2.1[B][2] Screenshot B**2.1[B][2] Screenshot C**

2.1[B][2] Screenshot D**2.1[B][3] – Lesson Dates_Archive Storage Lists****2.1[B][3] Screenshot A**

```

        ScheduledLessons_BookedTime.Add(DataReader.GetString(7));
        ScheduledLessons_EndDate.Add(DataReader.GetDateTime(8));

    }

}

// STORE LESSON DATE ARCHIVE INFORMATION
private void LocalStorage_LessonDate_Archive()
{
    using (connection = new SqlConnection(connectionString))
    {
        connection.Open();
        using (SqlCommand cmd = new SqlCommand("SELECT * FROM LessonDate_Archive", connection))
        {

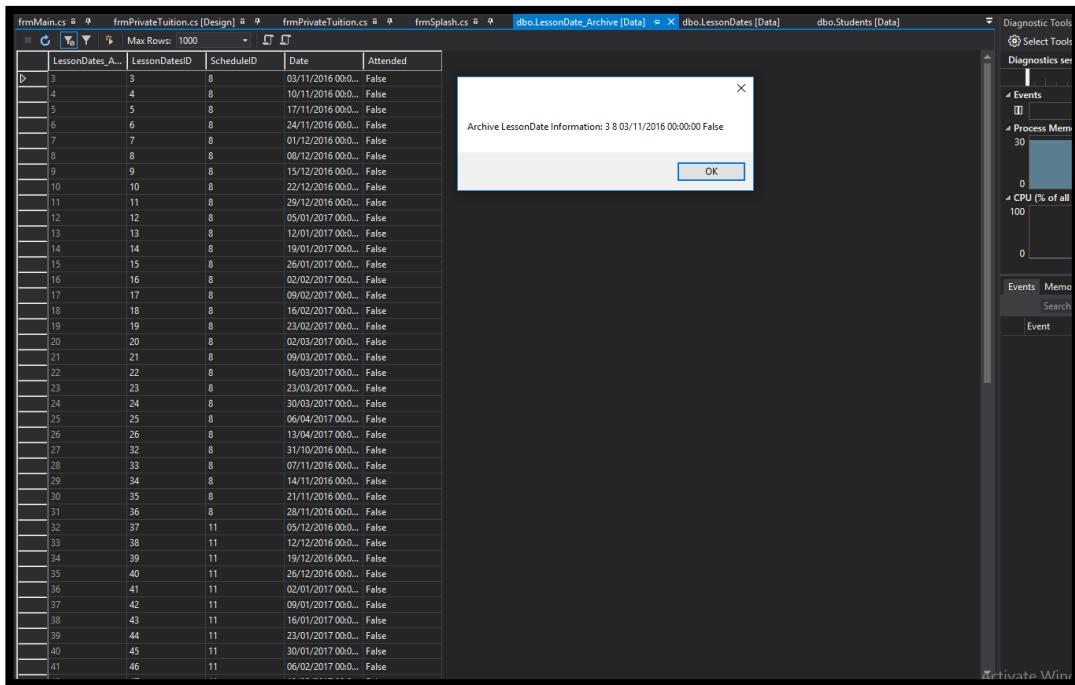
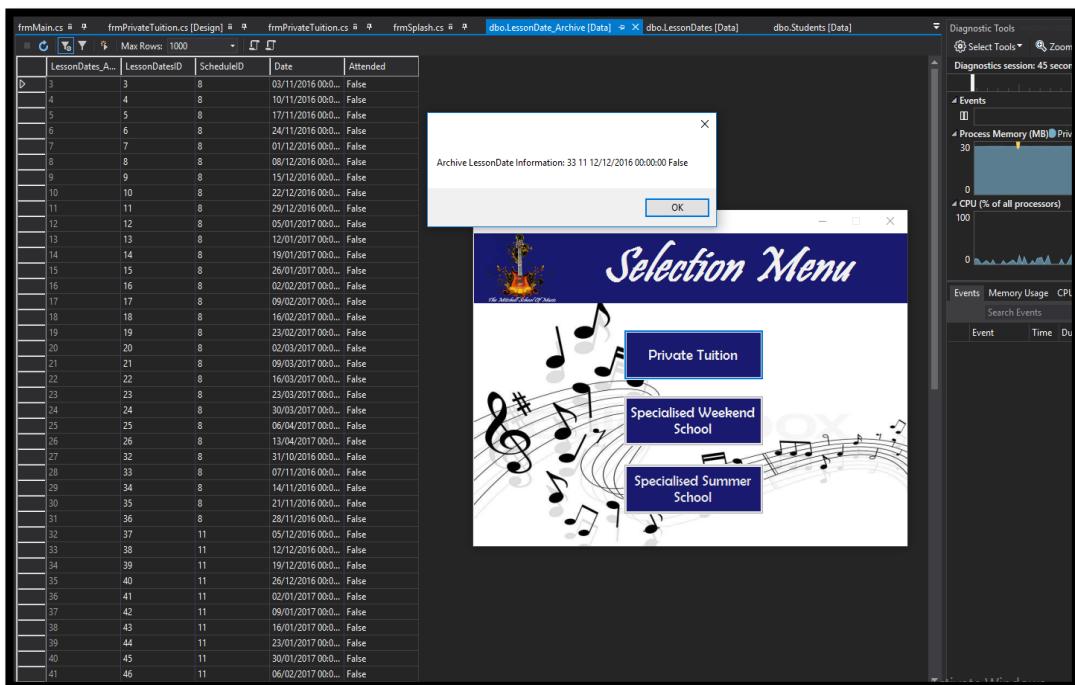
            DataReader = cmd.ExecuteReader();
            while (DataReader.Read())
            {
                LessonDate_Archive_ID.Add(DataReader.GetInt32(0));
                LessonDate_Archive_ScheduleID.Add(DataReader.GetInt32(1));
                LessonDate_Archive_Date.Add(DataReader.GetDateTime(3));
                LessonDate_Archive_Attended.Add(DataReader.GetBoolean(4));

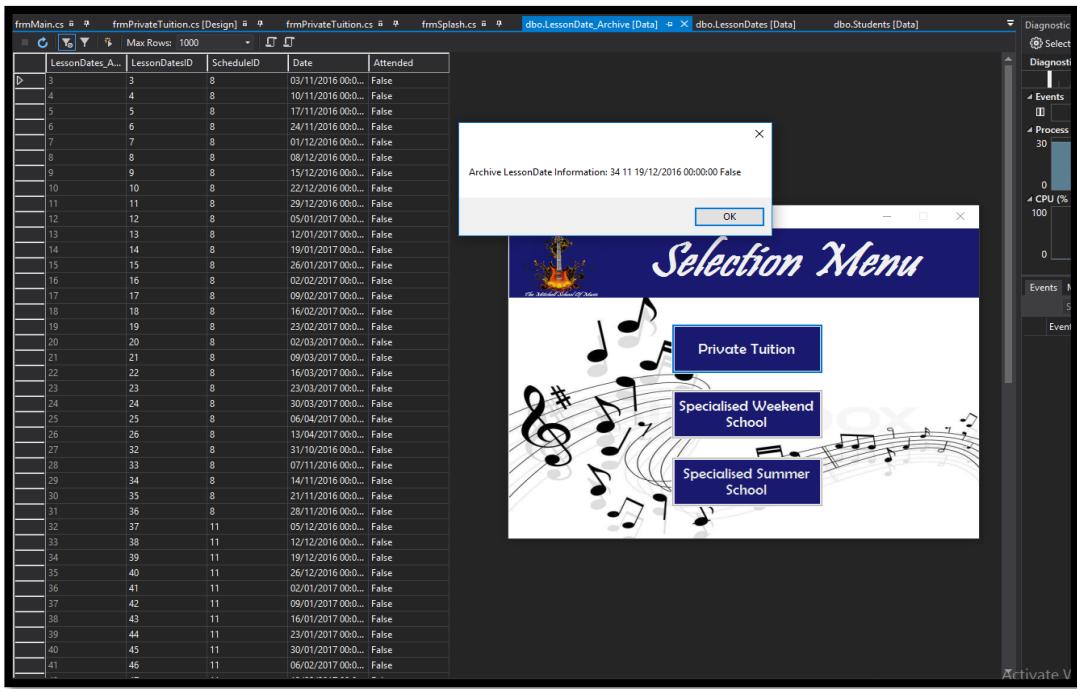
                Reallocation_LessonDates_ScheduleID.Add(DataReader.GetInt32(2));
                Reallocation_LessonDates_Dates.Add(DataReader.GetDateTime(5));
                Reallocation_LessonDates_Attended.Add(DataReader.GetBoolean(6));

                MessageBox.Show("Archive LessonDate Information: " + LessonDate_Archive_ID[test] + " " + LessonDate_Archive_ScheduleID[test] + " " + LessonDate_Archive_Date[test] + " " + LessonDate_Archive_Attended[test]);
                test++;
            }
        }
    }

// ARCHIVE COMPLETE LESSONS
private void Archive_Passed_LessonDates()
{
    if (Completed == false)
    {
        Completed = true;
    }
}

```

2.1[B][3] Screenshot B**2.1[B][3] Screenshot C**

2.1[B][3] Screenshot D**2.1[B][4] – Scheduled Lessons Storage Lists****2.1[B][4] Screenshot A**

```

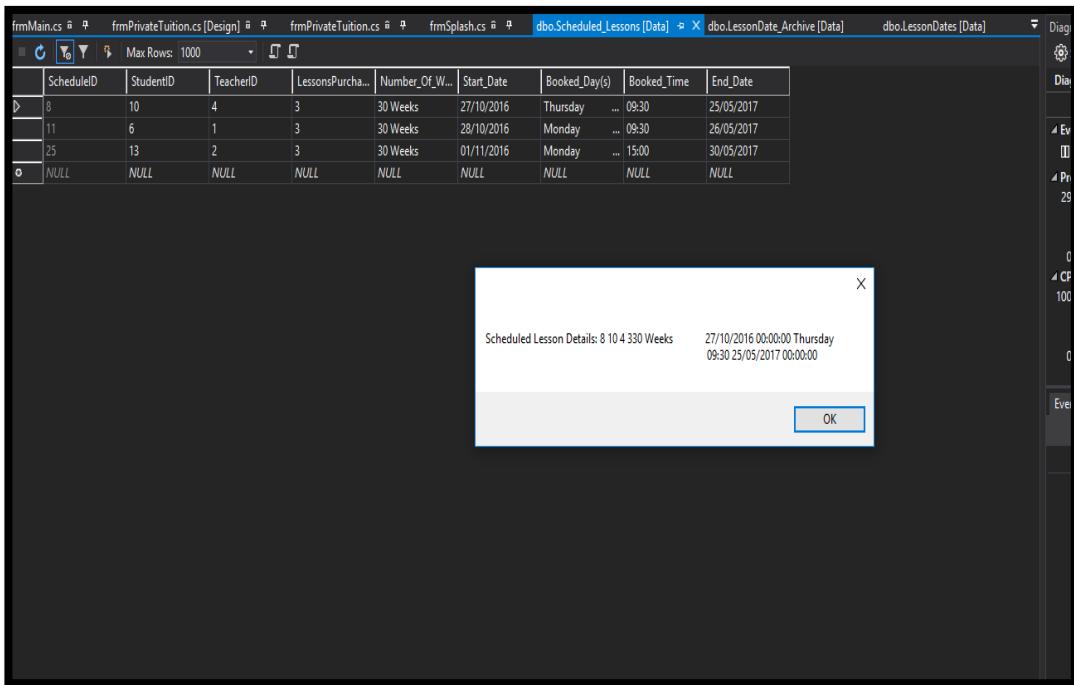
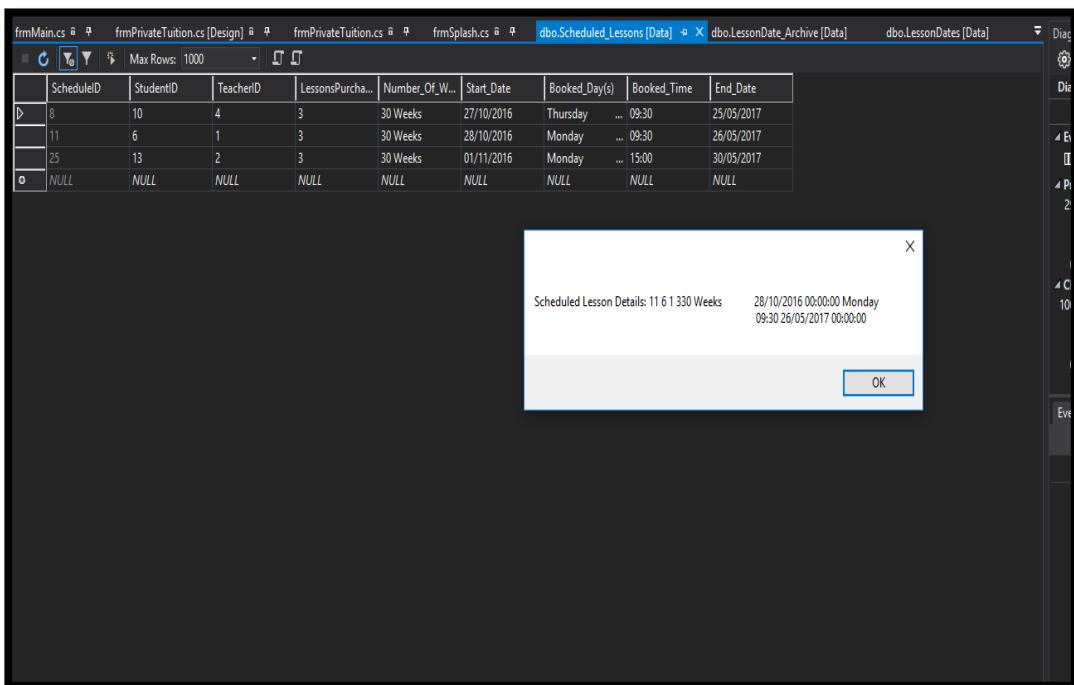
private void LocalStorage_SignedIn()
{
    Students.Payment.Add(DataReader.GetBoolean(12));
    Reallocation.StudentAbsences.Add(0);
}

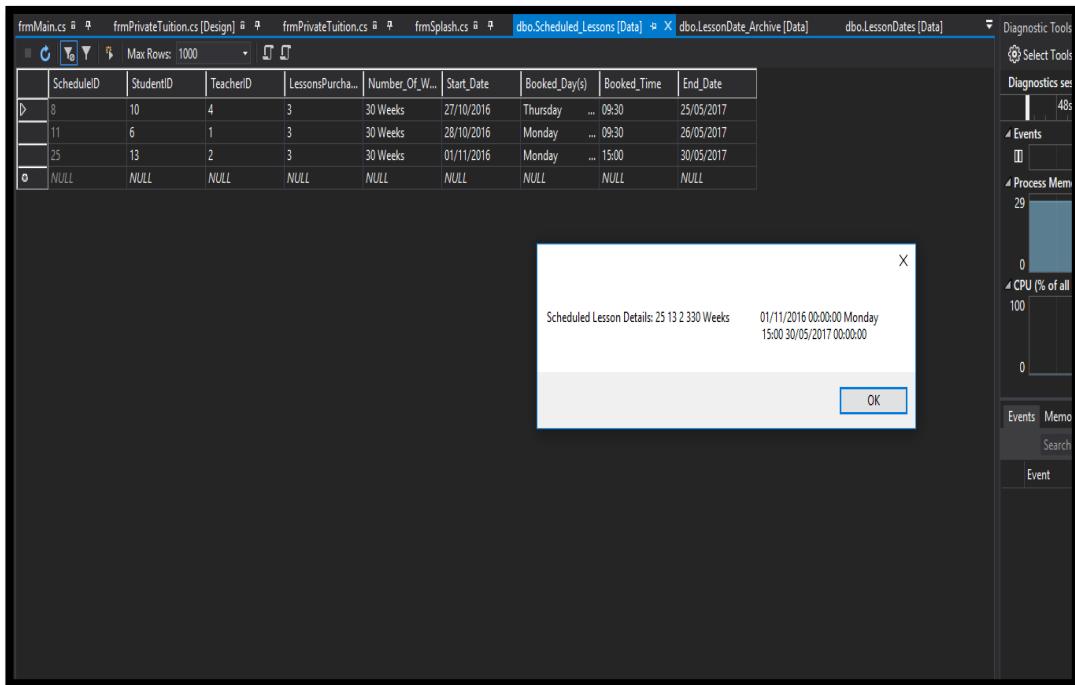
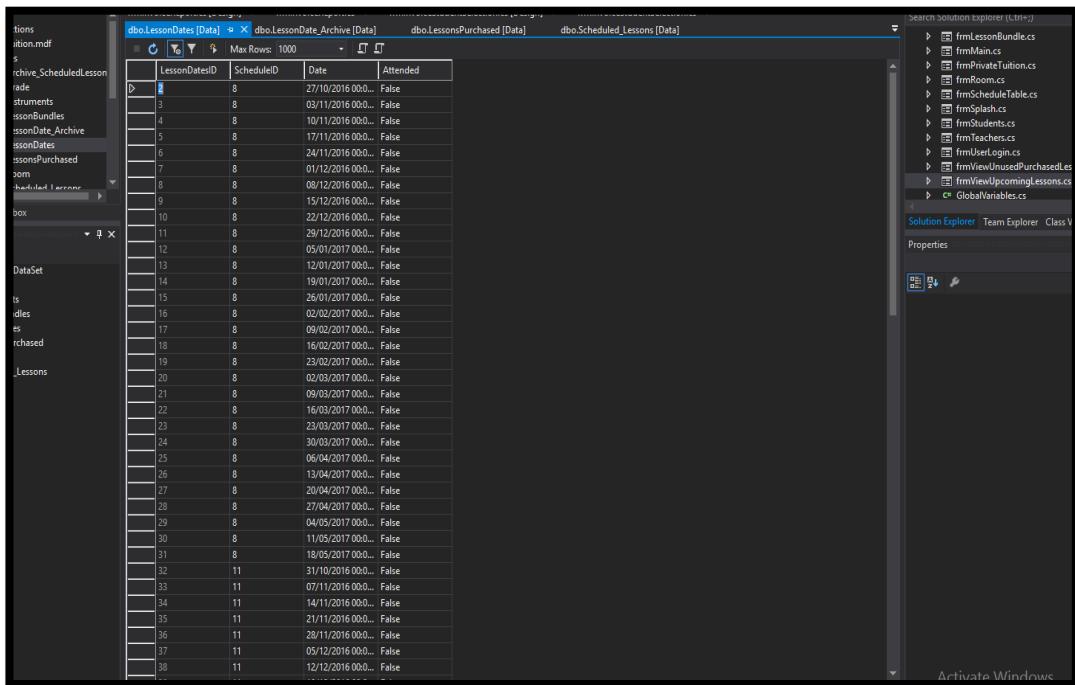
// STORE SCHEDULED LESSONS INFORMATION
private void LocalStorage_ScheduledLessons()
{
    using (connection = new SqlConnection(connectionString))
    {
        connection.Open();
        using (SqlCommand cmd = new SqlCommand("SELECT * FROM Scheduled_Lessons", connection))
        {
            DataReader = cmd.ExecuteReader();
            while (DataReader.Read())
            {
                ScheduledLessons_ID.Add(DataReader.GetInt32(0));
                ScheduledLessons_StudentID.Add(DataReader.GetInt32(1));
                ScheduledLessons_TeacherID.Add(DataReader.GetInt32(2));
                ScheduledLessons_StartDate.Add(DataReader.GetDateTime(3));
                ScheduledLessons_EndDate.Add(DataReader.GetDateTime(4));
                ScheduledLessons_NumberOfWeeks.Add(DataReader.GetInt32(5));
                ScheduledLessons_StartDate.Add(DataReader.GetDateTime(6));
                ScheduledLessons_BookedDays.Add(DataReader.GetString(7));
                ScheduledLessons_BookedTime.Add(DataReader.GetString(8));
                ScheduledLessons_EndDate.Add(DataReader.GetDateTime(9));

                MessageBox.Show("Scheduled Lesson Details: " + ScheduledLessons_ID[test] + " " + ScheduledLessons_StudentID[test] + " " + ScheduledLessons_TeacherID[test] + " " + ScheduledLessons_PurchasedLessonID[test] +
                               " " + ScheduledLessons_NumberOfWeeks[test] + " " + ScheduledLessons_StartDate[test] + " " + ScheduledLessons_BookedDays[test] + " " + ScheduledLessons_BookedTime[test] + " " + ScheduledLessons_EndDate[test]);
            }
            test++;
        }
    }
}

// STORE LESSON DATE ARCHIVE INFORMATION
private void LocalStorage_LessonDate_Archive()
{
    using (connection = new SqlConnection(connectionString))
    {
        connection.Open();
        using (SqlCommand cmd = new SqlCommand("SELECT * FROM LessonDate_Archive", connection))
        {
            DataReader = cmd.ExecuteReader();
        }
    }
}

```

2.1[B][4] Screenshot B**2.1[B][4] Screenshot C**

2.1[B][4] Screenshot D**Process [C]) LessonDates Archive****2.1[C][1] – Archived Lesson Dates****2.1[C][1] Screenshot A**

2.1[C][1] Screenshot B

	LessonDatesID	LessonDateID	ScheduleID	Date	Attended
0	NULL	NULL	NULL	NULL	NULL

2.1[C][1] Screenshot C

	LessonDatesID	ScheduleID	Date	Attended
0	8	8	27/04/2017 00:00:00	False
29	8	8	04/05/2017 00:00:00	False
30	8	8	11/05/2017 00:00:00	False
31	8	8	18/05/2017 00:00:00	False
58	11	11	01/05/2017 00:00:00	False
59	11	11	08/05/2017 00:00:00	False
60	11	11	15/05/2017 00:00:00	False
61	11	11	22/05/2017 00:00:00	False
87	25	25	01/05/2017 00:00:00	False
88	25	25	08/05/2017 00:00:00	False
89	25	25	15/05/2017 00:00:00	False
90	25	25	22/05/2017 00:00:00	False
91	25	25	29/05/2017 00:00:00	False
0	NULL	NULL	NULL	NULL

2.1[C][1] Screenshot D

LessonDates_A... e	LessonDatesID 1	ScheduledID 1	Date 1	Attended 1
44	48	11	20/02/2017 00:00...	False
45	49	11	27/02/2017 00:00...	False
46	50	11	06/03/2017 00:00...	False
47	51	11	13/03/2017 00:00...	False
48	52	11	20/03/2017 00:00...	False
49	53	11	27/03/2017 00:00...	False
50	54	11	03/04/2017 00:00...	False
51	55	11	10/04/2017 00:00...	False
52	56	11	17/04/2017 00:00...	False
53	57	11	24/04/2017 00:00...	False
54	62	11	07/11/2016 00:00...	False
55	63	11	14/11/2016 00:00...	False
56	64	11	21/11/2016 00:00...	False
57	65	11	28/11/2016 00:00...	False
58	66	11	05/12/2016 00:00...	False
59	67	11	12/12/2016 00:00...	False
60	68	11	19/12/2016 00:00...	False
61	69	11	26/12/2016 00:00...	False
62	70	25	02/01/2017 00:00...	False
63	71	25	09/01/2017 00:00...	False
64	72	25	16/01/2017 00:00...	False
65	73	25	23/01/2017 00:00...	False
66	74	25	30/01/2017 00:00...	False
67	75	25	06/02/2017 00:00...	False
68	76	25	13/02/2017 00:00...	False
69	77	25	20/02/2017 00:00...	False
70	78	25	27/02/2017 00:00...	False
71	79	25	06/03/2017 00:00...	False
72	80	25	13/03/2017 00:00...	False
73	81	25	20/03/2017 00:00...	False
74	82	25	27/03/2017 00:00...	False
75	83	25	03/04/2017 00:00...	False
76	84	25	10/04/2017 00:00...	False
77	85	25	17/04/2017 00:00...	False
78	86	25	24/04/2017 00:00...	False
o	NULL	NULL	NULL	NULL

Process [D]) Clear Old LessonDate Archives

2.1[D][1] – Lesson Date Archived Records**2.1[D][1] Screenshot A**

LessonDates_A... e	LessonDatesID 1	ScheduledID 1	Date 1	Attended 1
44	48	11	20/02/2017 00:00...	False
45	49	11	27/02/2017 00:00...	False
46	50	11	06/03/2017 00:00...	False
47	51	11	13/03/2017 00:00...	False
48	52	11	20/03/2017 00:00...	False
49	53	11	27/03/2017 00:00...	False
50	54	11	03/04/2017 00:00...	False
51	55	11	10/04/2017 00:00...	False
52	56	11	17/04/2017 00:00...	False
53	57	11	24/04/2017 00:00...	False
54	62	11	07/11/2016 00:00...	False
55	63	11	14/11/2016 00:00...	False
56	64	11	21/11/2016 00:00...	False
57	65	11	28/11/2016 00:00...	False
58	66	11	05/12/2016 00:00...	False
59	67	11	12/12/2016 00:00...	False
60	68	11	19/12/2016 00:00...	False
61	69	11	26/12/2016 00:00...	False
62	70	25	02/01/2017 00:00...	False
63	71	25	09/01/2017 00:00...	False
64	72	25	16/01/2017 00:00...	False
65	73	25	23/01/2017 00:00...	False
66	74	25	30/01/2017 00:00...	False
67	75	25	06/02/2017 00:00...	False
68	76	25	13/02/2017 00:00...	False
69	77	25	20/02/2017 00:00...	False
70	78	25	27/02/2017 00:00...	False
71	79	25	06/03/2017 00:00...	False
72	80	25	13/03/2017 00:00...	False
73	81	25	20/03/2017 00:00...	False
74	82	25	27/03/2017 00:00...	False
75	83	25	03/04/2017 00:00...	False
76	84	25	10/04/2017 00:00...	False
77	85	25	17/04/2017 00:00...	False
78	86	25	24/04/2017 00:00...	False
o	NULL	NULL	NULL	NULL

Activate Windows

2.1[D][1] Screenshot B

LessonDates_A...	LessonDatesID	ScheduledID	Date	Attended
43	47	11	13/02/2017 00:00:00	False
44	48	11	20/02/2017 00:00:00	False
45	49	11	27/02/2017 00:00:00	False
46	50	11	06/03/2017 00:00:00	False
47	51	11	13/03/2017 00:00:00	False
48	52	11	20/03/2017 00:00:00	False
49	53	11	27/03/2017 00:00:00	False
50	54	11	03/04/2017 00:00:00	False
51	55	11	10/04/2017 00:00:00	False
52	56	11	17/04/2017 00:00:00	False
53	57	11	24/04/2017 00:00:00	False
54	62	11	07/11/2016 00:00:00	False
55	63	11	14/11/2016 00:00:00	False
56	64	11	21/11/2016 00:00:00	False
57	65	11	28/11/2016 00:00:00	False
58	66	11	05/12/2016 00:00:00	False
59	67	11	12/12/2016 00:00:00	False
60	68	11	19/12/2016 00:00:00	False
61	69	11	26/12/2016 00:00:00	False
62	70	25	02/01/2017 00:00:00	False
63	71	25	09/01/2017 00:00:00	False
64	72	25	16/01/2017 00:00:00	False
65	73	25	23/01/2017 00:00:00	False
66	74	25	30/01/2017 00:00:00	False
67	75	25	06/02/2017 00:00:00	False
68	76	25	13/02/2017 00:00:00	False
69	77	25	20/02/2017 00:00:00	False
70	78	25	27/02/2017 00:00:00	False
71	79	25	06/03/2017 00:00:00	False
72	80	25	13/03/2017 00:00:00	False
73	81	25	20/03/2017 00:00:00	False
74	82	25	27/03/2017 00:00:00	False
75	83	25	03/04/2017 00:00:00	False
76	84	25	10/04/2017 00:00:00	False
77	85	25	17/04/2017 00:00:00	False
78	86	25	10/10/2015 00:00:00	False
o	NULL	NULL	NULL	NULL

2.1[D][1] Screenshot C

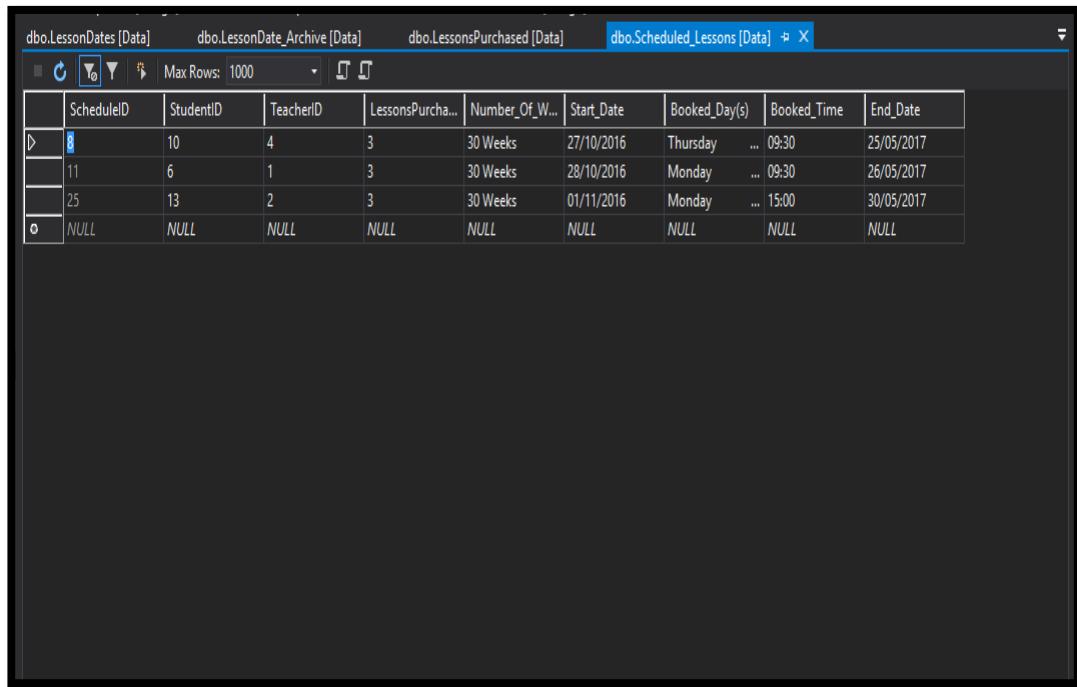
LessonDates_A...	LessonDatesID	ScheduledID	Date	Attended
20	20	8	02/03/2017 00:00:00	False
21	21	8	09/03/2017 00:00:00	False
22	22	8	16/03/2017 00:00:00	False
23	23	8	23/03/2017 00:00:00	False
24	24	8	30/03/2017 00:00:00	False
25	25	8	06/04/2017 00:00:00	False
26	26	8	13/04/2017 00:00:00	False
27	27	8	20/04/2017 00:00:00	False
45	49	11	27/02/2017 00:00:00	False
46	50	11	06/03/2017 00:00:00	False
47	51	11	13/03/2017 00:00:00	False
48	52	11	20/03/2017 00:00:00	False
49	53	11	27/03/2017 00:00:00	False
50	54	11	03/04/2017 00:00:00	False
51	55	11	10/04/2017 00:00:00	False
52	56	11	17/04/2017 00:00:00	False
53	57	11	24/04/2017 00:00:00	False
70	78	25	27/02/2017 00:00:00	False
71	79	25	06/03/2017 00:00:00	False
72	80	25	13/03/2017 00:00:00	False
73	81	25	20/03/2017 00:00:00	False
74	82	25	27/03/2017 00:00:00	False
75	83	25	03/04/2017 00:00:00	False
76	84	25	10/04/2017 00:00:00	False
77	85	25	17/04/2017 00:00:00	False
o	NULL	NULL	NULL	NULL

Process [G] Archive Scheduled Lessons***2.1[G][1] – Archived Scheduled Lesson Records******2.1[G][1] Screenshot A***

	ScheduleID	StudentID	TeacherID	LessonsPurcha...	Number_Of_W...	Start_Date	Booked_Day(s)	Booked_Time	End_Date
	8	10	4	3	30 Weeks	27/10/2016	Thursday	... 09:30	25/05/2017
	11	6	1	3	30 Weeks	28/10/2016	Monday	... 09:30	26/05/2017
	25	13	2	3	30 Weeks	01/11/2016	Monday	... 15:00	30/05/2017
	50	6	3	13	10 Weeks	26/10/2016	Tuesday	... 14:00	20/03/2016
✖	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

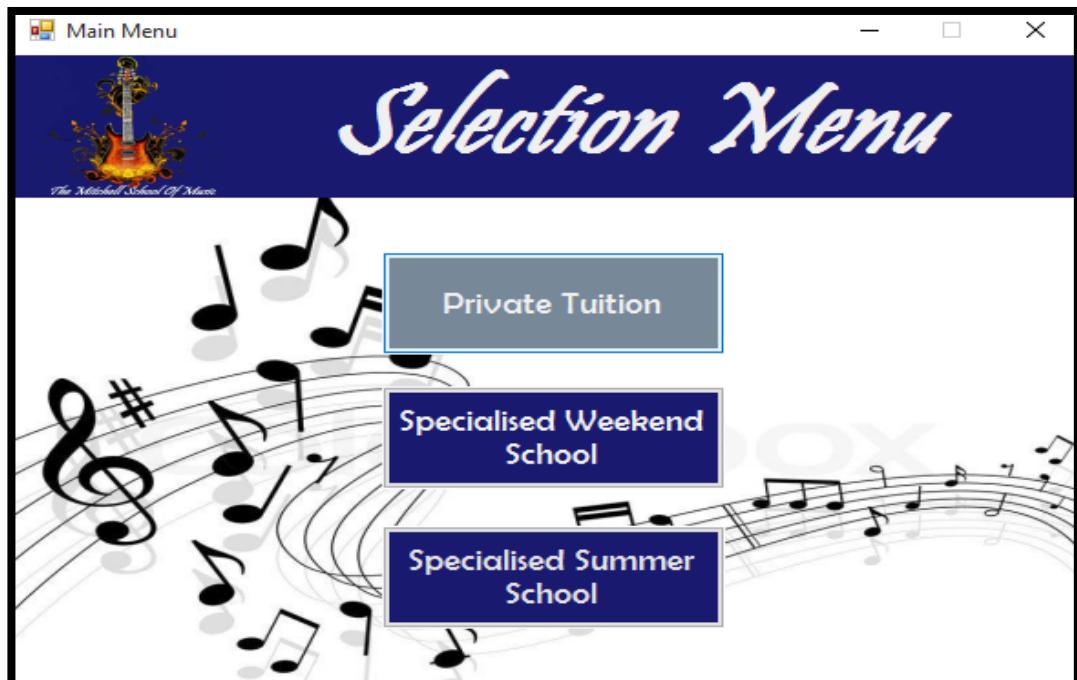
2.1[G][1] Screenshot B

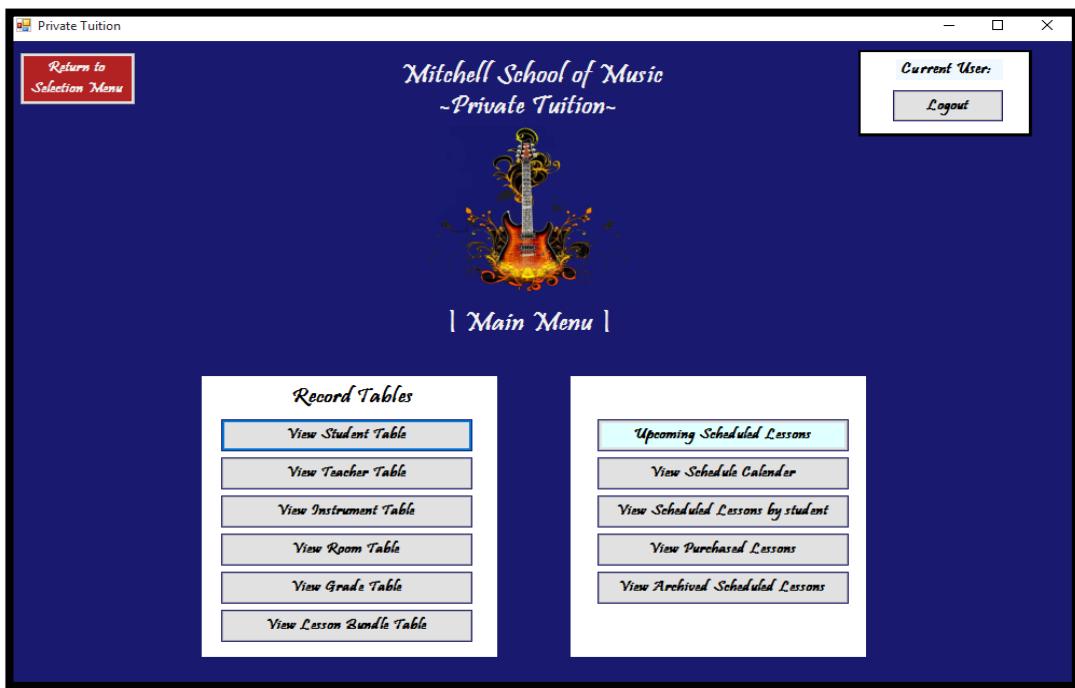
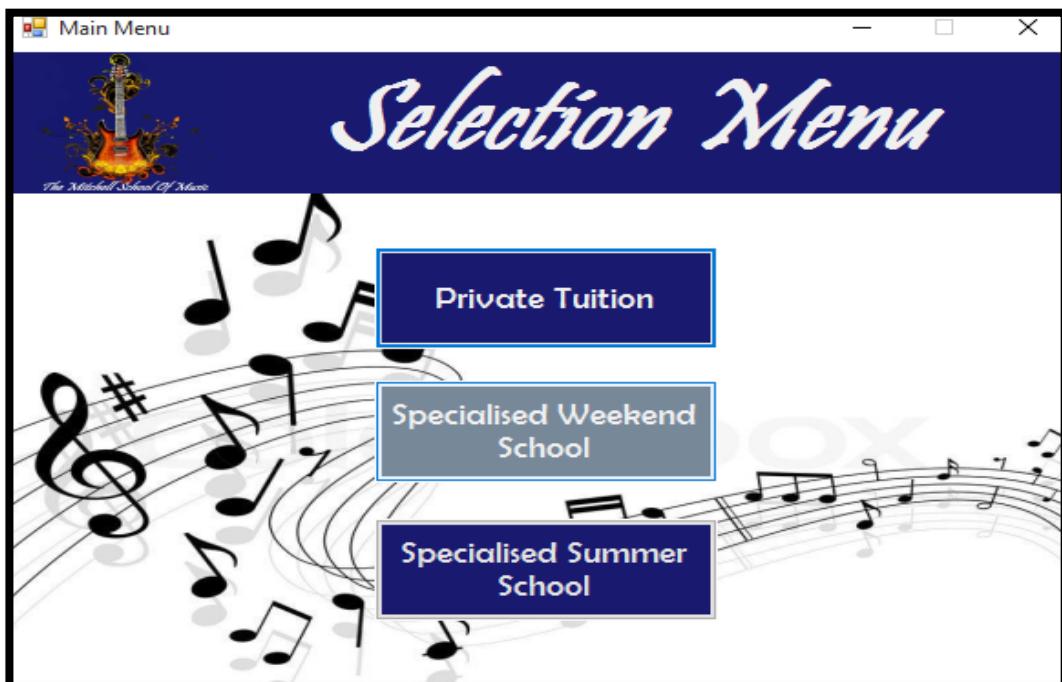
	Schedule_Arch...	ScheduleID	StudentID	TeacherID	Purchased_Les...	Number_of_W...	Start_Date	Booked_Days	Booked_Time	End_Date
▶	1	6	7	2	1	10 Weeks	26/10/2016 00:0...	Monday	... 09:30:00	04/01/2017 00:0...
	2	7	7	5	2	10 Weeks	24/10/2016 00:0...	Tuesday + Wed...	09:30:00	02/01/2017 00:0...
	3	12	16	1	1	5 Weeks	25/10/2016 00:0...	Monday + Frid...	11:30:00	29/11/2016 00:0...
	4	13	12	3	1	10 Weeks	24/10/2016 00:0...	Tuesday	... 12:00:00	02/01/2017 00:0...
	5	14	22	2	2	15 Weeks	28/10/2016 00:0...	Thursday + Frid...	13:00:00	10/02/2017 00:0...
	6	15	5	3	1	5 Weeks	25/10/2016 00:0...	Monday + Tues...	14:30:00	29/11/2016 00:0...
	7	16	17	4	2	15 Weeks	27/10/2016 00:0...	Wednesday + T...	16:00:00	09/02/2017 00:0...
	8	17	4	2	1	10 Weeks	05/01/2017 00:0...	Friday	... 10:30:00	04/01/2017 00:0...
	9	18	8	3	2	20 Weeks	28/10/2016 00:0...	Thursday	... 11:00:00	17/03/2017 00:0...
	10	19	5	2	2	20 Weeks	24/10/2016 00:0...	Friday	... 13:00:00	13/03/2017 00:0...
	11	20	21	3	1	5 Weeks	26/10/2016 00:0...	Monday + Tues...	12:00:00	30/11/2016 00:0...
	12	21	19	3	1	10 Weeks	28/10/2016 00:0...	Wednesday	... 13:00:00	06/01/2017 00:0...
	13	22	6	1	1	10 Weeks	26/10/2016 00:0...	Thursday	... 10:00:00	04/01/2017 00:0...
	14	23	10	2	2	20 Weeks	03/11/2016 00:0...	Friday	... 14:00:00	23/03/2017 00:0...
	15	24	8	1	2	15 Weeks	02/11/2016 00:0...	Wednesday + T...	15:00:00	15/02/2017 00:0...
	16	27	27	3	1	5 Weeks	03/11/2016 00:0...	Tuesday + Frid...	14:00:00	08/12/2016 00:0...
	17	28	6	5	2	15 Weeks	14/11/2016 00:0...	Monday + Tues...	13:00:00	27/02/2017 00:0...
	18	29	18	3	1	5 Weeks	15/11/2016 00:0...	Wednesday + F...	13:00:00	20/12/2016 00:0...
	19	30	15	3	2	20 Weeks	21/11/2016 00:0...	Monday + Tues...	14:00:00	10/04/2017 00:0...
	20	31	13	2	3	20 Weeks	01/09/2017 00:0...	Tuesday + Wed...	13:30:00	01/01/2017 00:0...
	21	32	12	5	2	15 Weeks	01/09/2017 00:0...	Monday + Frid...	14:00:00	14/01/2017 00:0...
	22	33	19	3	1	10 Weeks	01/09/2017 00:0...	Tuesday	... 16:30:00	10/04/2017 00:0...
	23	34	22	2	3	25 Weeks	01/09/2017 00:0...	Friday	... 09:30:00	10/03/2017 00:0...
✖	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	

2.1[G][1] Screenshot C


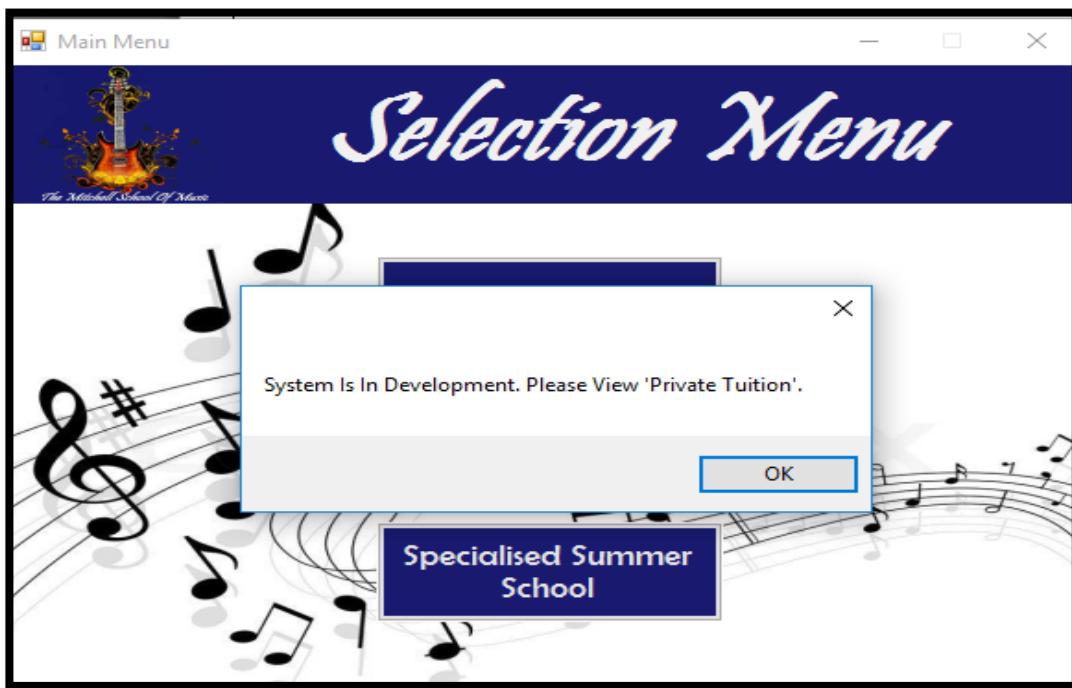
The screenshot shows a Microsoft SQL Server Management Studio window displaying a table named 'dbo.Scheduled_Lessons'. The table has the following columns: ScheduleID, StudentID, TeacherID, LessonsPurcha..., Number_Of_W..., Start_Date, Booked_Day(s), Booked_Time, and End_Date. There are four rows of data:

ScheduleID	StudentID	TeacherID	LessonsPurcha...	Number_Of_W...	Start_Date	Booked_Day(s)	Booked_Time	End_Date
10	4	3	30 Weeks	27/10/2016	Thursday	... 09:30	25/05/2017	
11	6	1	3	30 Weeks	28/10/2016	Monday	... 09:30	26/05/2017
25	13	2	3	30 Weeks	01/11/2016	Monday	... 15:00	30/05/2017
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

2.2) Main Menu*Process [A]) Navigational Buttons****2.2[A][1] – Private Tuition Button******2.2[A][1] Screenshot A***

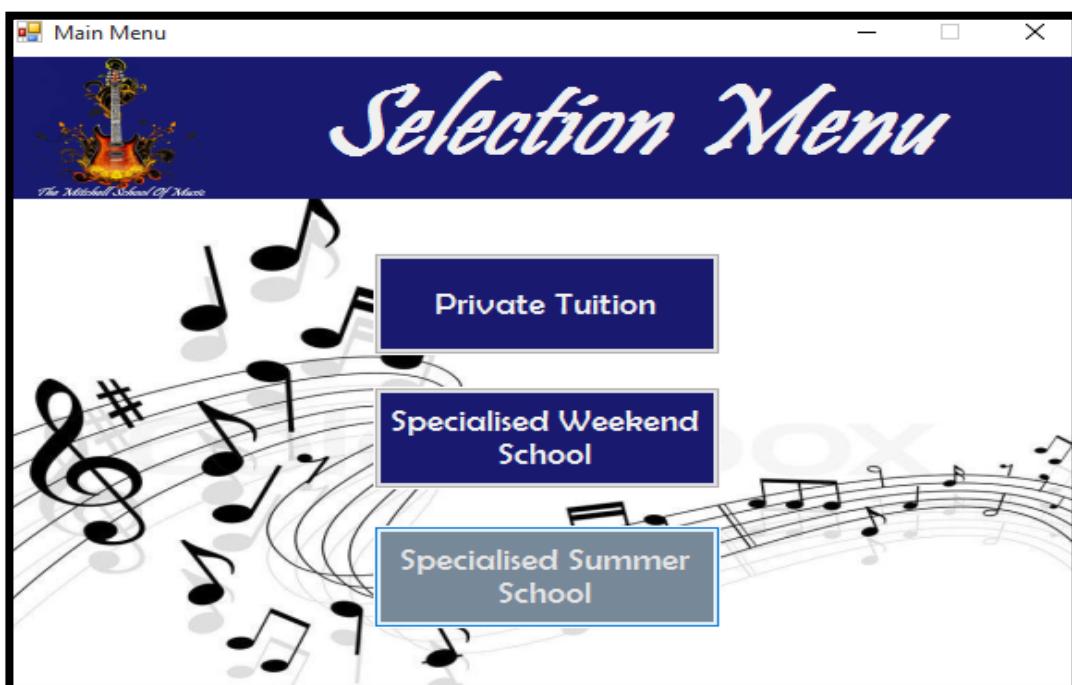
2.2[A][1] Screenshot B**2.2[A][2] – Specialist Weekend School****2.2[A][2] Screenshot A**

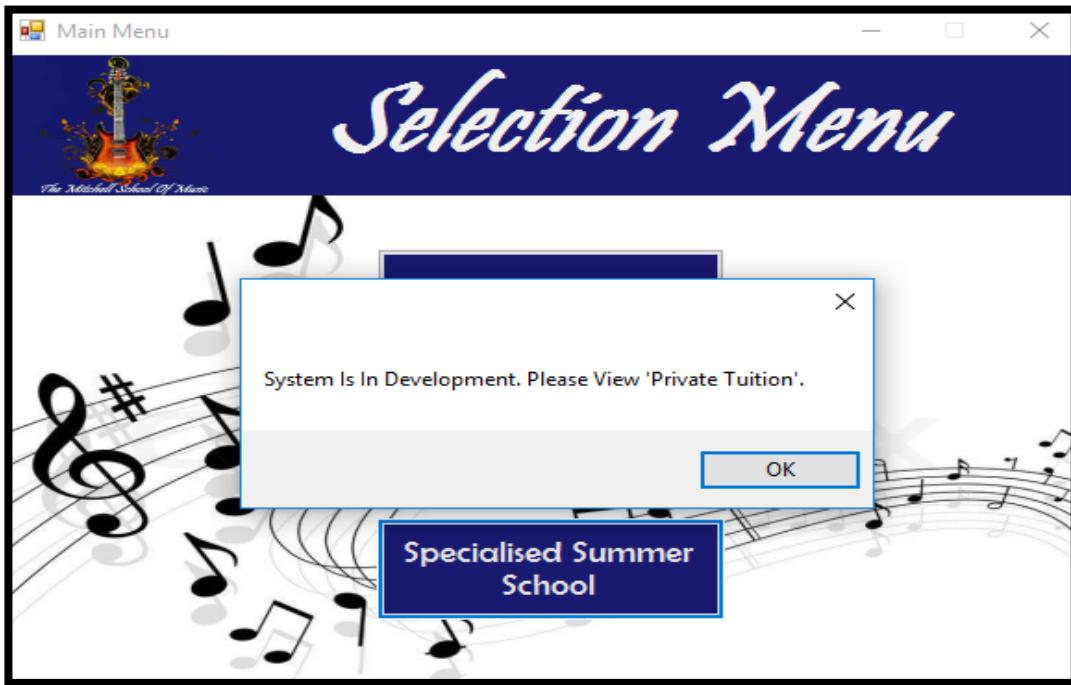
2.2[A][2] Screenshot B



2.2[A][3] – Specialist Summer School

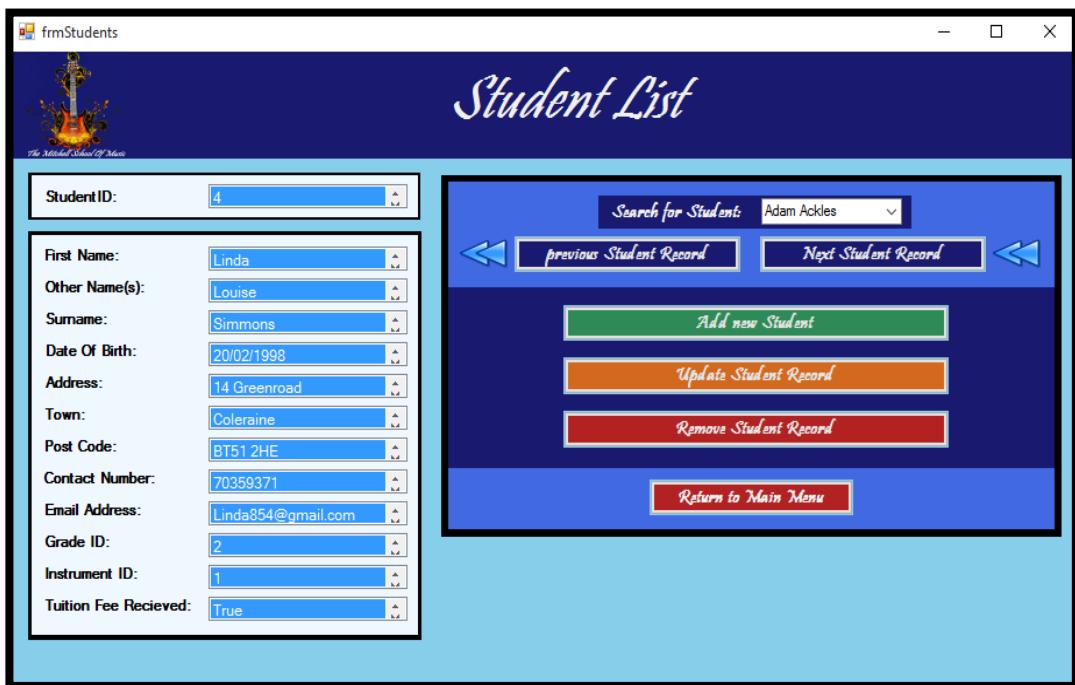
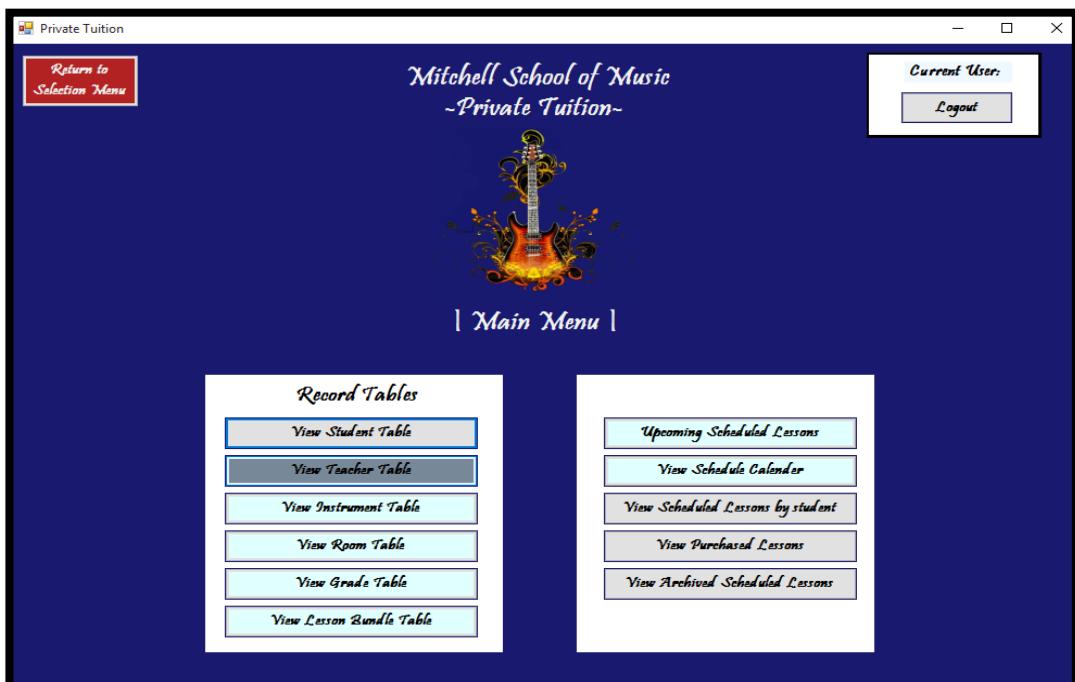
2.2[A][3] Screenshot A

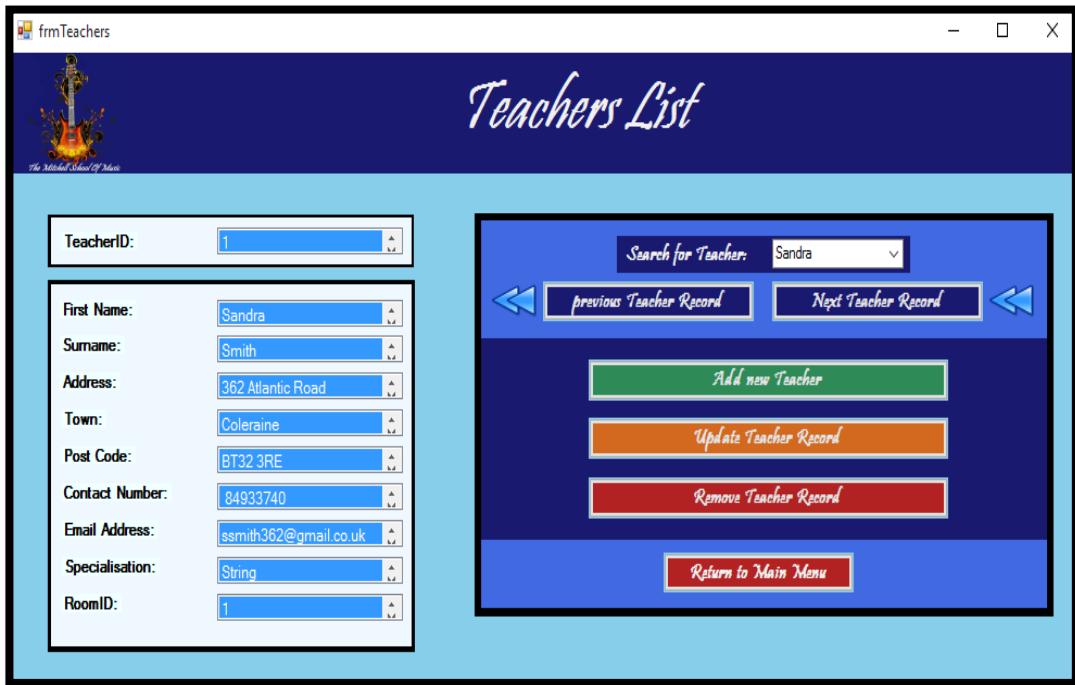


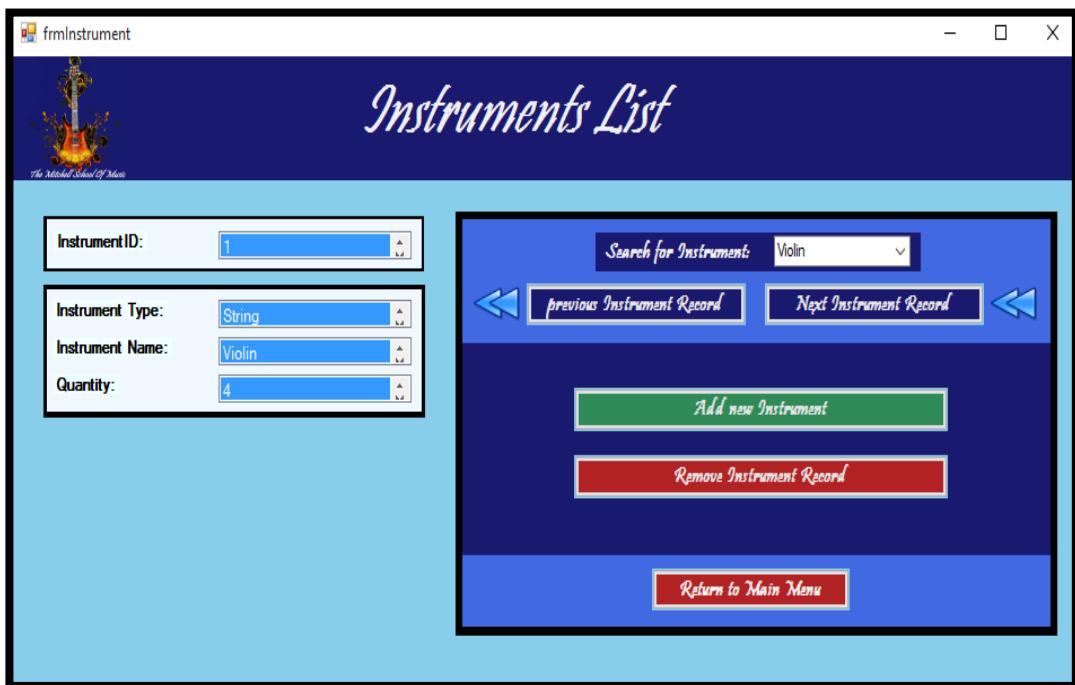
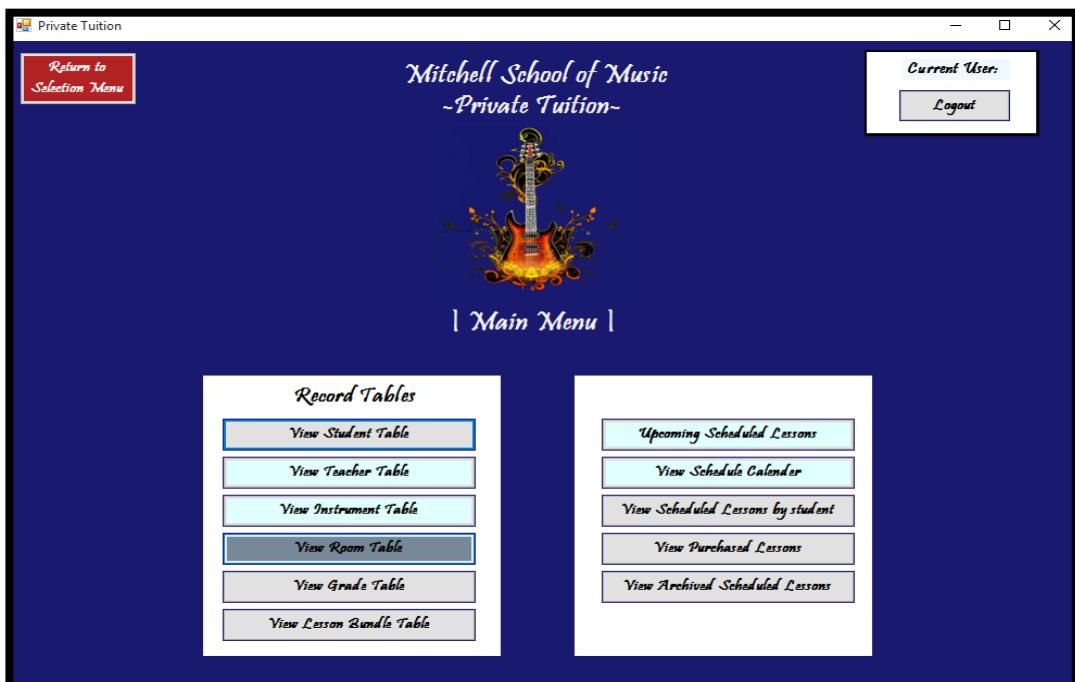
2.2[A][3] Screenshot B**2.3) Private Tuition Menu**

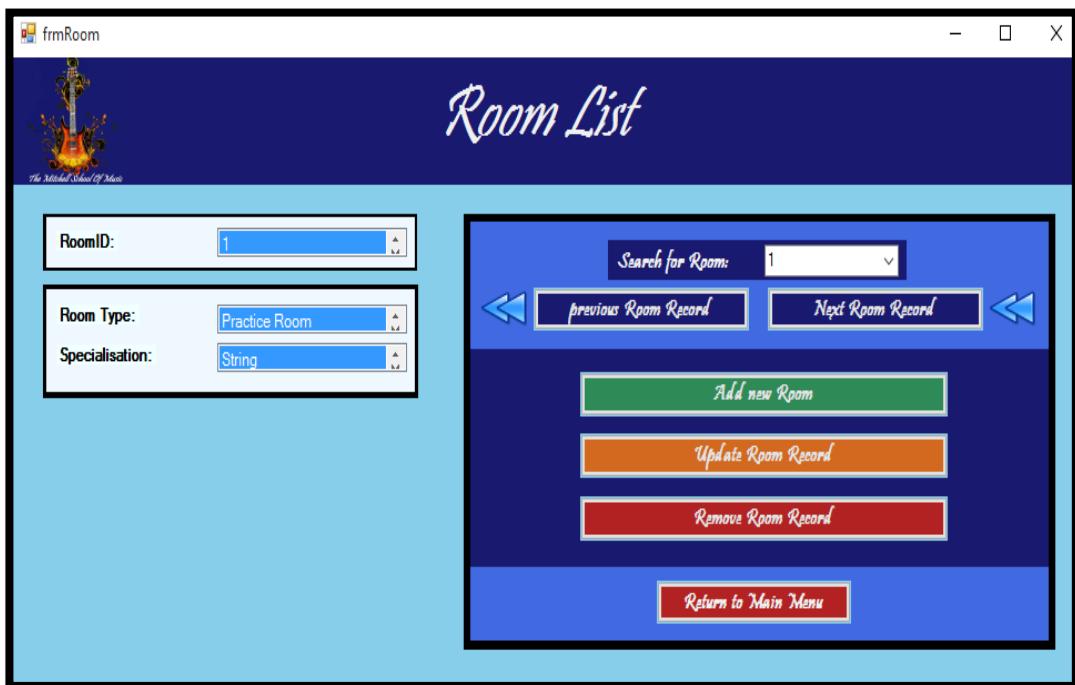
Process [A]) Navigational Buttons

2.3[A][1] – Students Table**2.3[A][1] Screenshot A**

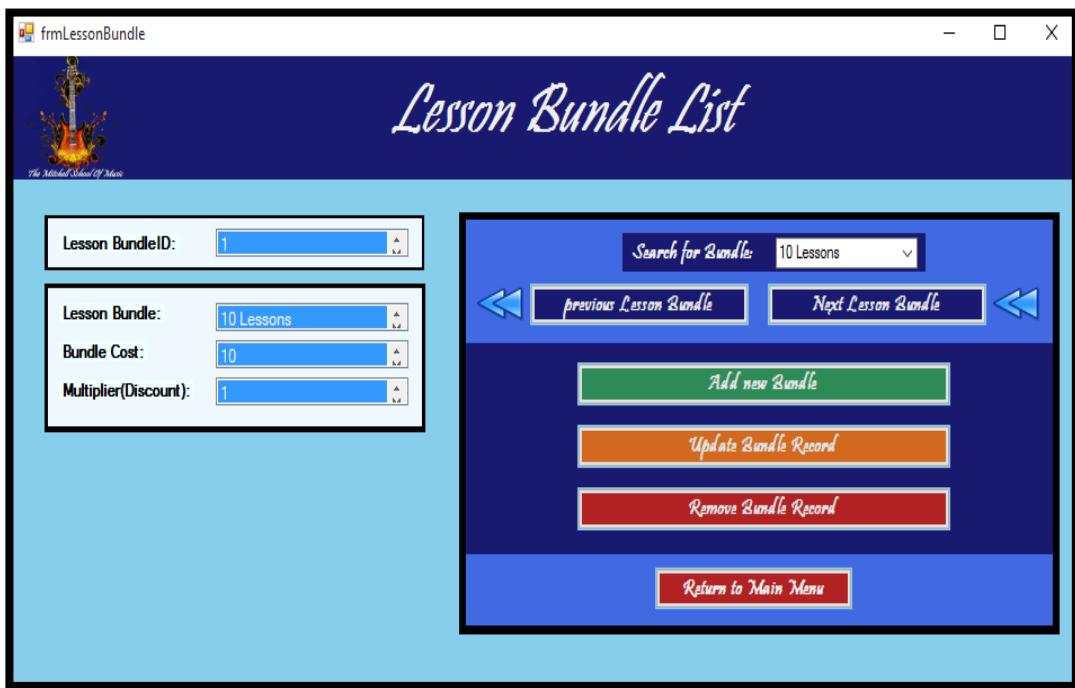
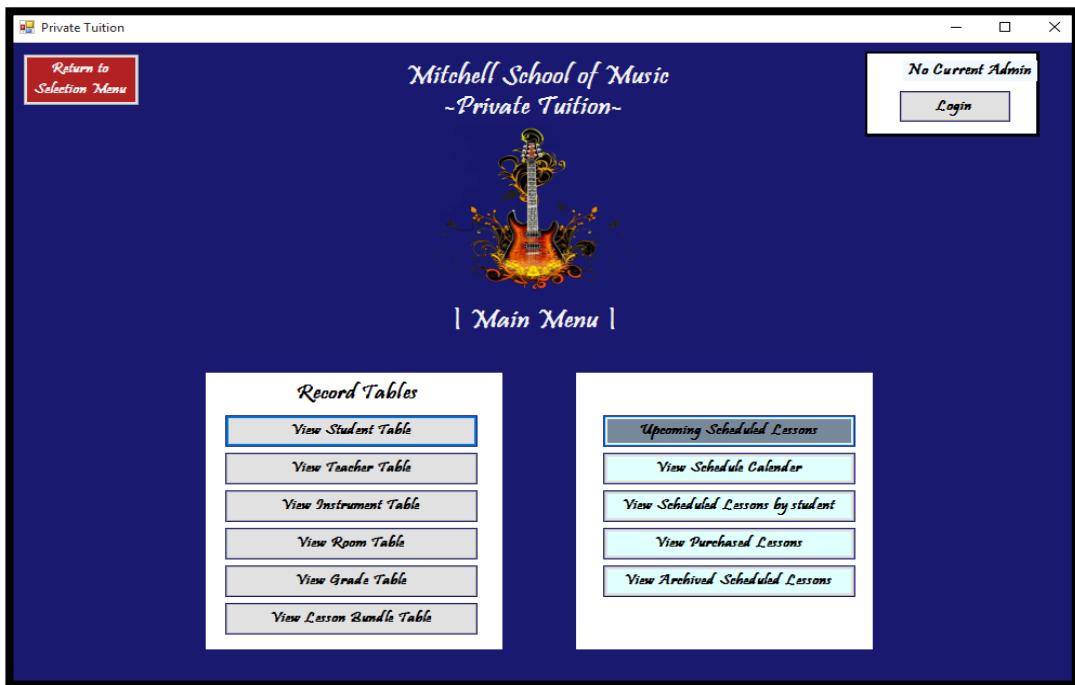
2.3[A][1] Screenshot B**2.3[A][2] – Teachers Table****2.3[A][2] Screenshot A**

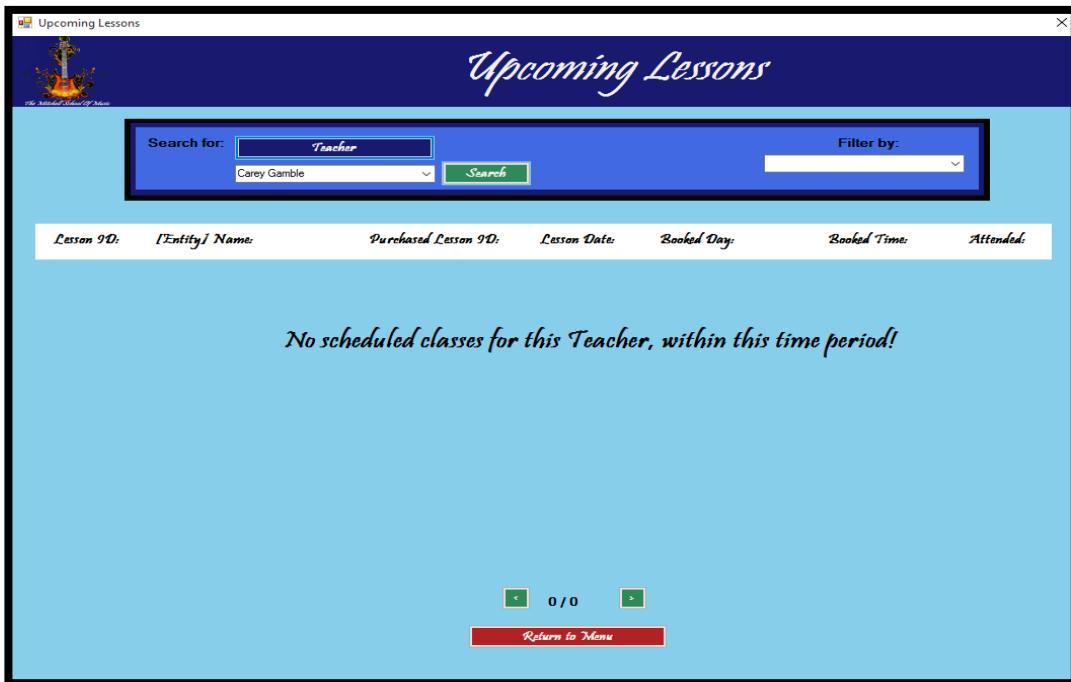
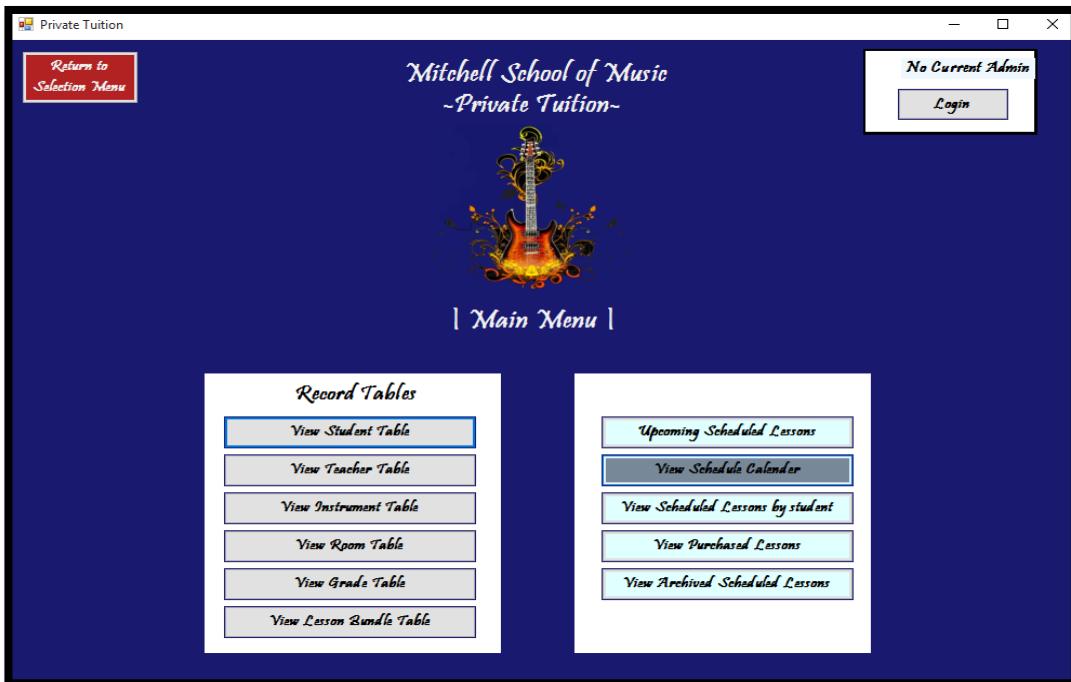
2.3[A][2] Screenshot B**2.3[A][3] – Instrument Table****2.3[A][3] Screenshot A**

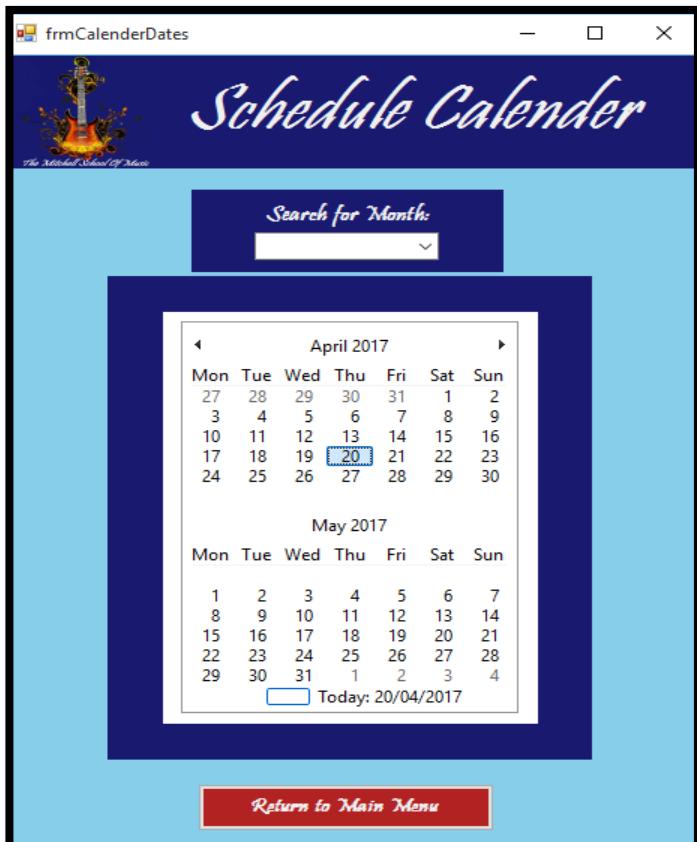
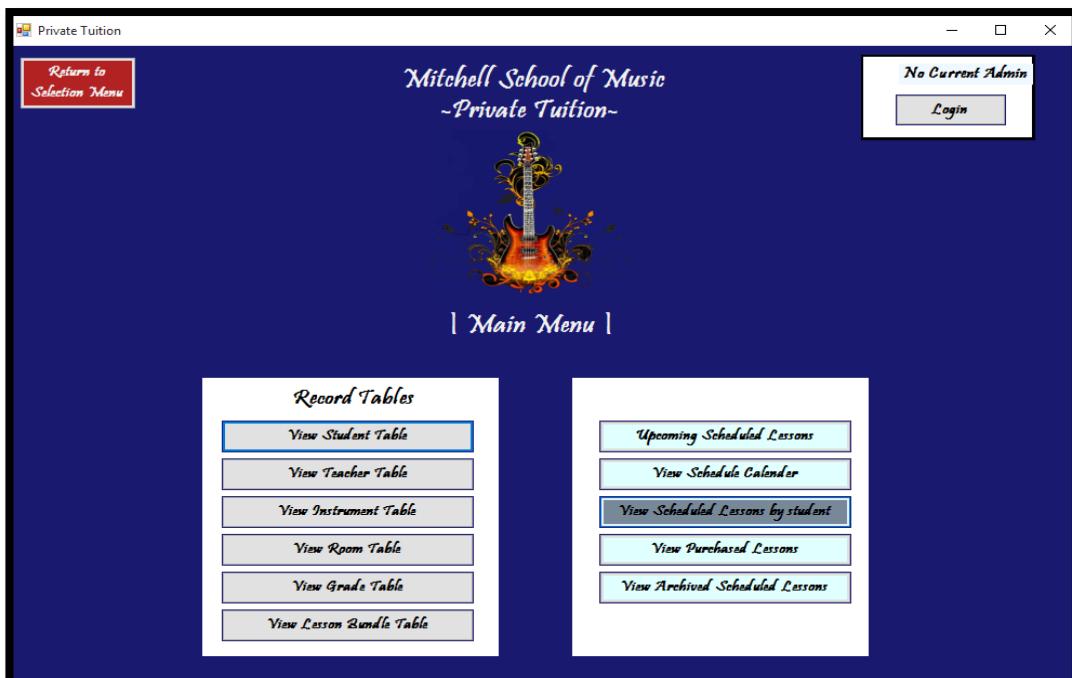
2.3[A][3] Screenshot B**2.3[A][4] – Room Table****2.3[A][4] Screenshot A**

2.3[A][4] Screenshot B**2.3[A][5] – Grade Table****2.3[A][5] Screenshot A**

2.3[A][5] Screenshot B**2.3[A][6] – Lesson Bundle Table****2.3[A][6] Screenshot A**

2.3[A][6] Screenshot B**2.3[A][9] – Upcoming Scheduled Lessons Table****2.3[A][9] Screenshot A**

2.3[A][9] Screenshot B**2.3[A][10] – Schedule Calendar****2.3[A][10] Screenshot A**

2.3[A][10] Screenshot B**2.3[A][11] – Scheduled Lessons Table****2.3[A][11] Screenshot A**

2.3[A][11] Screenshot B

Scheduled Lessons List

StudentID: <input type="text" value="6"/>	TeacherID: <input type="text" value="1"/>	Search for Student: Adam Ackles previous Student Record Next Student Record																												
Student Details First Name: <input type="text" value="Daniel"/> Surname: <input type="text" value="Waters"/> Contact Number: <input type="text" value="37719048"/>		Teacher Details First Name: <input type="text" value="Sandra"/> Surname: <input type="text" value="Smith"/> Specialisation: <input type="text" value="String"/> Room: <input type="text" value="1"/>																												
Grade Details GradeID: <input type="text" value="4"/> Grade: <input type="text" value="Diploma"/>		Add new Student Update Lesson Record Remove Lesson Record Return to Calendar [Placeholder] Return to Main Menu																												
ScheduleID: <input type="text" value="11"/> <table border="1"> <thead> <tr> <th>ScheduleID</th> <th>StudentID</th> <th>TeacherID</th> <th>LessonsPurchasedID</th> <th>Number_of_Weeks</th> <th>Start_Date</th> <th>Booked_Day(s)</th> <th>Booked_Time</th> <th>End_Date</th> </tr> </thead> <tbody> <tr> <td>11</td> <td>6</td> <td>1</td> <td>3</td> <td>30 Weeks</td> <td>28/10/2016</td> <td>Monday</td> <td>09:30</td> <td>26/05/2017</td> </tr> <tr> <td>*</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>				ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date	11	6	1	3	30 Weeks	28/10/2016	Monday	09:30	26/05/2017	*								
ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date																						
11	6	1	3	30 Weeks	28/10/2016	Monday	09:30	26/05/2017																						
*																														

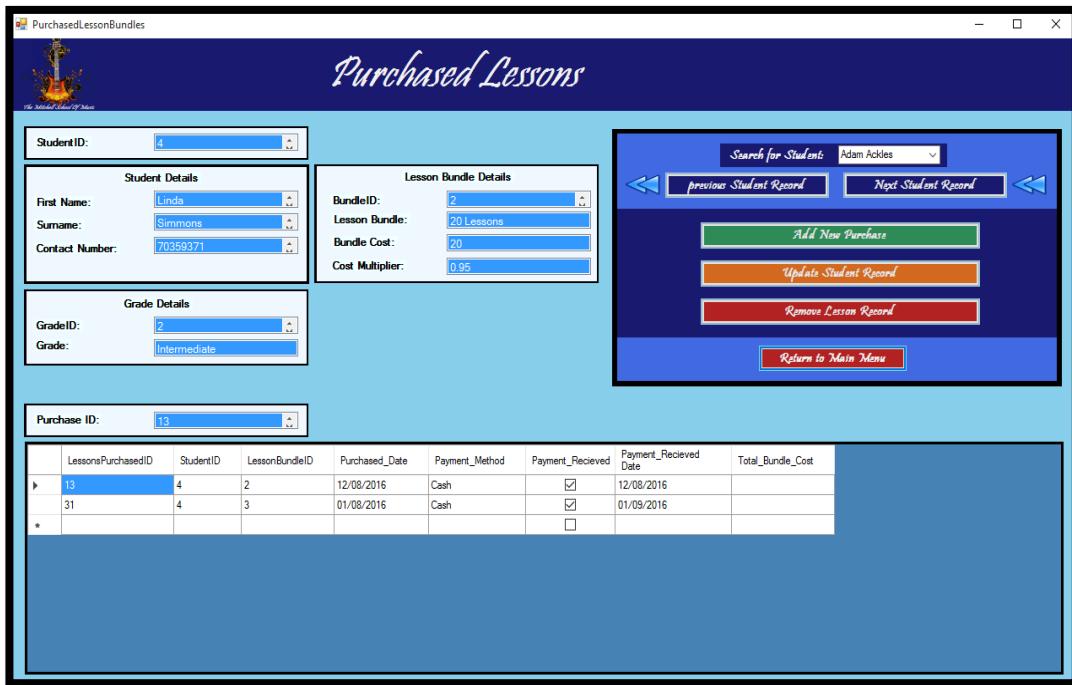
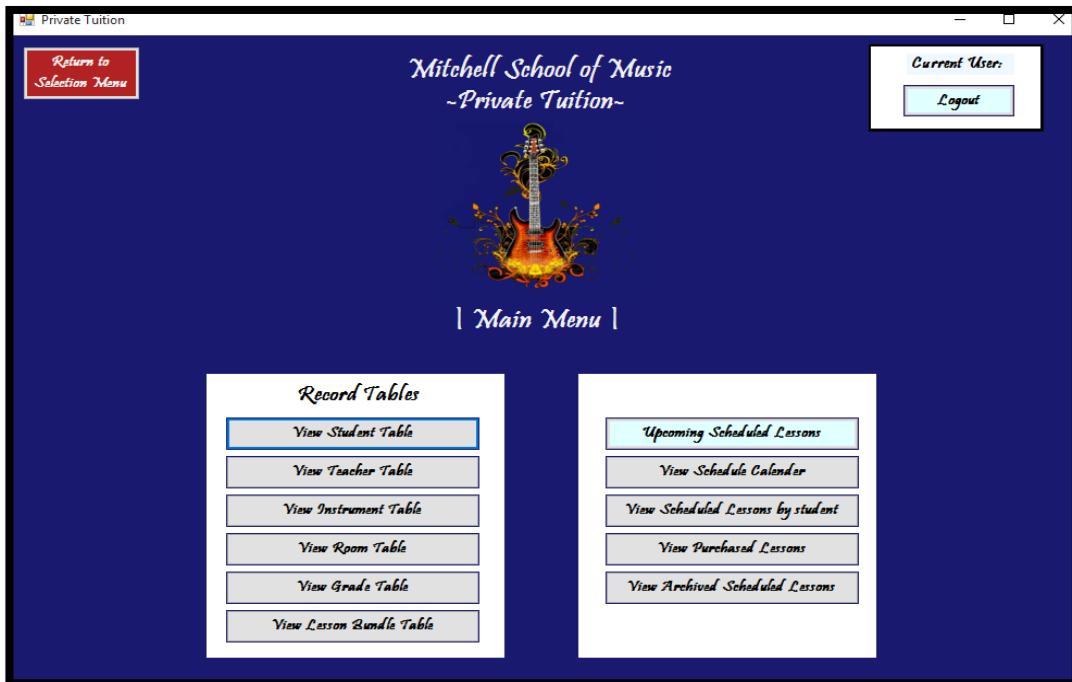
2.3[A][12] – Purchased Lessons Table**2.3[A][12] Screenshot A**

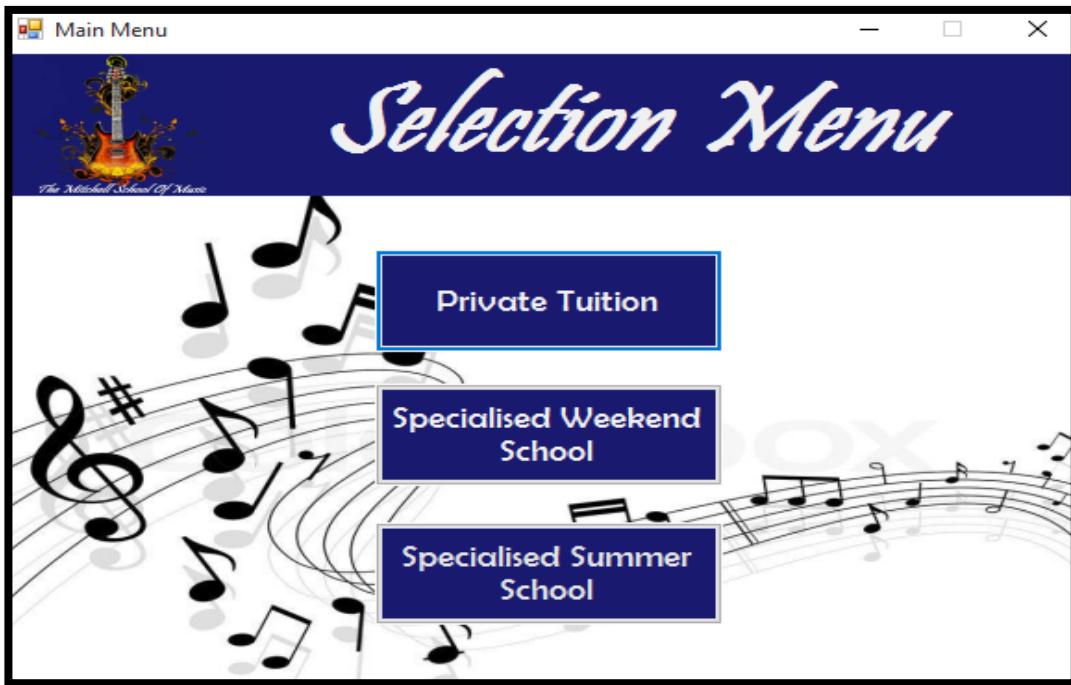
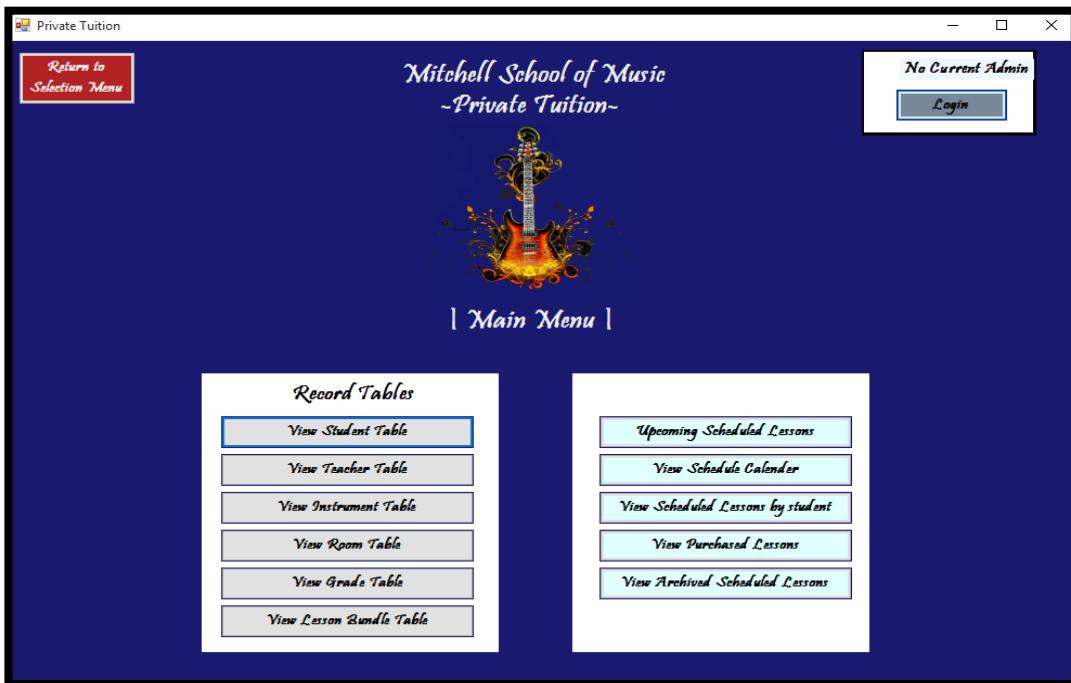
Mitchell School of Music
~Private Tuition~

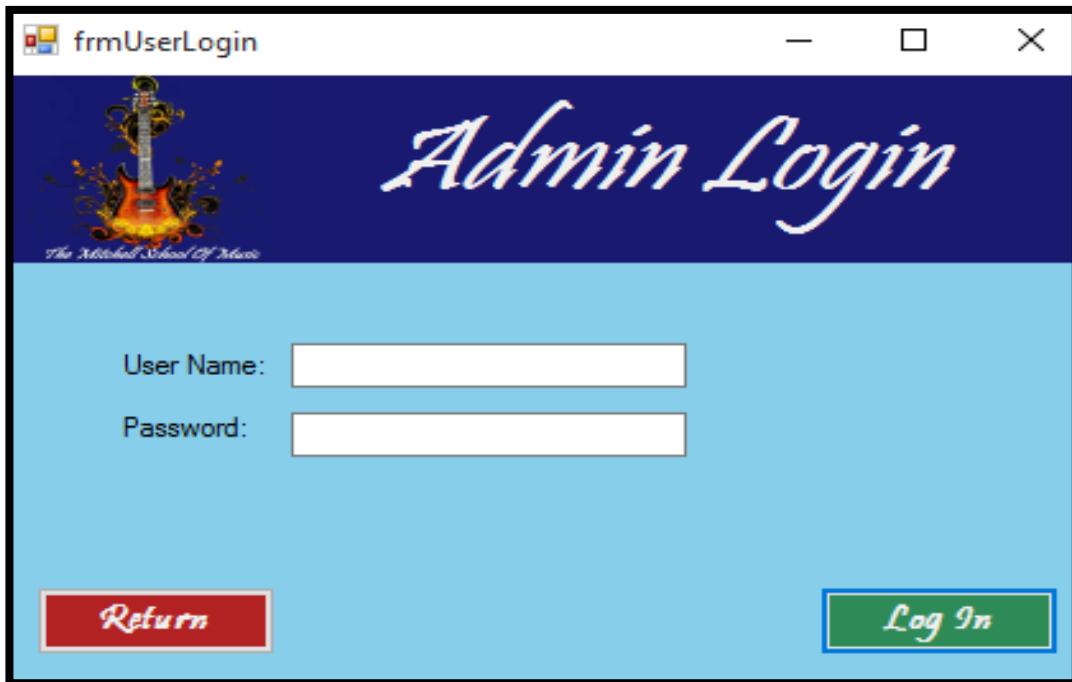
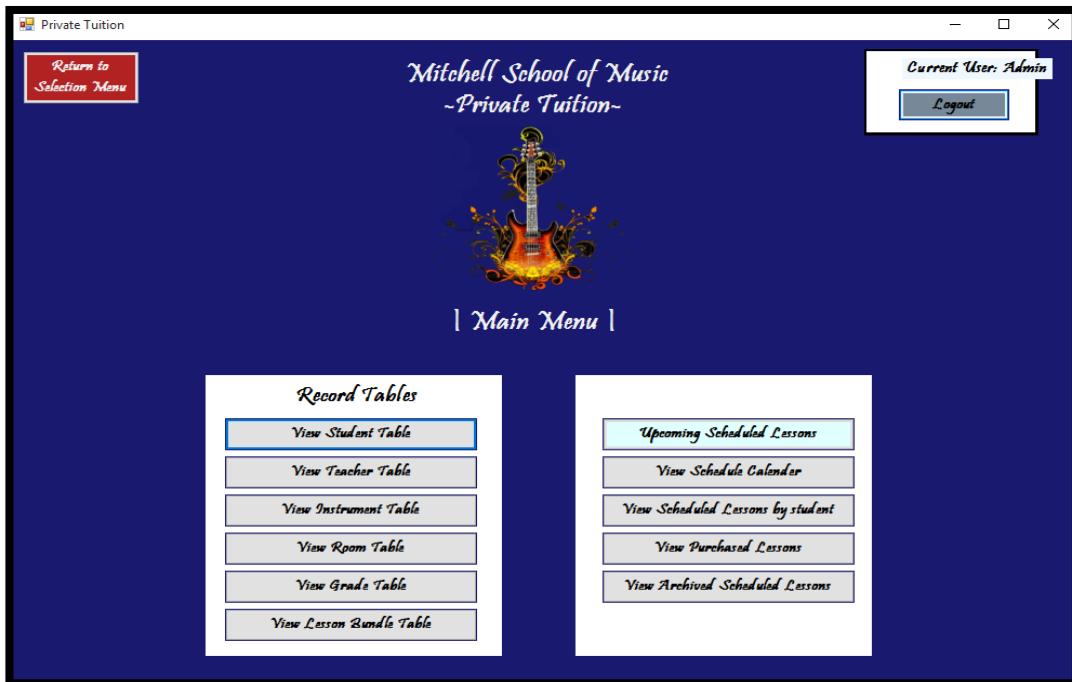
Main Menu

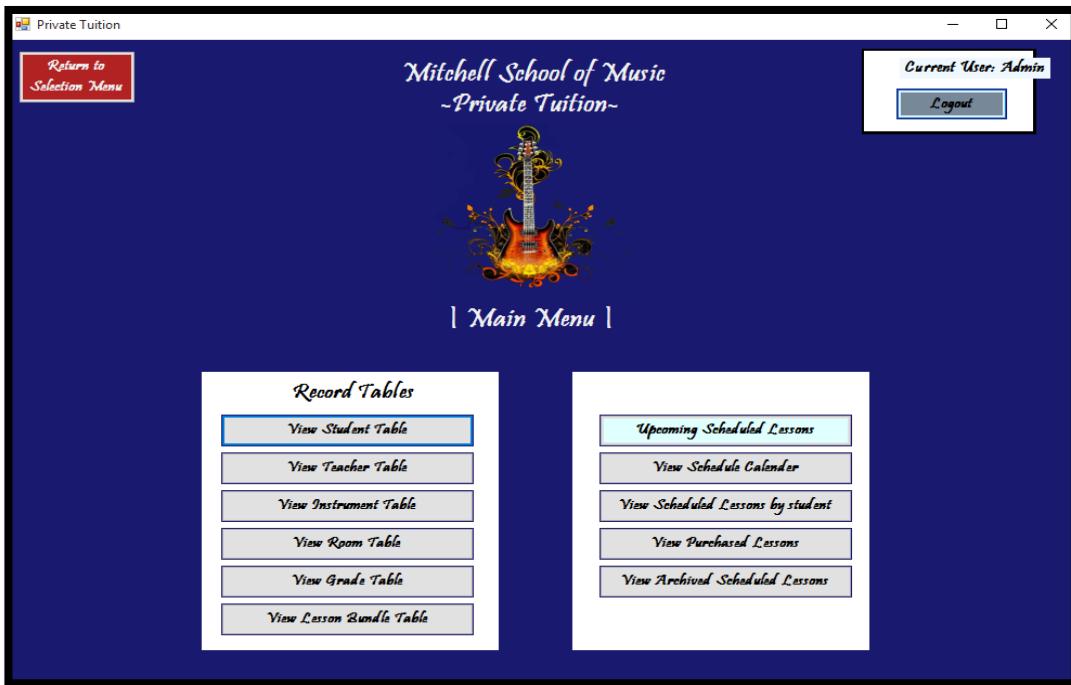
Record Tables View Student Table View Teacher Table View Instrument Table View Room Table View Grade Table View Lesson Bundle Table		Upcoming Scheduled Lessons View Schedule Calendar View Scheduled Lessons by student View Purchased Lessons View Archived Scheduled Lessons
---	--	--

No Current Admin
[Login](#)

2.3[A][12] Screenshot B**2.3[A][13] – Return to Selection Menu****2.3[A][13] Screenshot A**

2.3[A][13] Screenshot B**Process [B]) Login Function****2.3[B][1] – Login {Select} [No user Logged In]****2.3[B][1] Screenshot A**

2.3[B][1] Screenshot B**2.3[B][2] – Login {Select} [User Logged In]****2.3[B][2] Screenshot A**

2.3[B][2] Screenshot B**2.3[B][3] – Login {Display} + Login Message****2.3[B][3] Screenshot A**

2.3[B][3] Screenshot B**2.3[B][4] – User Label****2.3[B][4] Screenshot A**

2.3[B][4] Screenshot B**2.4) Students Form***Process [A]) Information Display***2.4[A][1] – Student Information****2.4[A][1] Screenshot A**

StudentID	First_Name	Other_Name(s)	Surname	DateOfBirth	Address	Town	PostCode	Contact_Num...	Email_Address	GradeID	InstrumentID	Tuition_Fee_R...
1	Linda	Louise	Simmons	20/02/1998	14 Greenroad	Coleraine	BT51 2HE	70359371	Linda54@gmail...	2	1	True
5	Trevor	Luke	Simmons	20/08/1997	22 The Glebe	Ballymoney	BT32 3ND	74920033	TrevorSimmons...	1	4	True
6	Daniel	Robert	Waters	01/02/1970	199 Pacradio	Ballymoney	BT47 3MF	37719048	danielwaters9...	4	3	True
7	Adam	Brent	Ackles	30/06/1993	731 boathane	Gavagh	BT43 2HT	46299567	adambrent@ya...	2	2	False
8	Jenson	James	Tweed	22/12/1990	71 glebe avenue	Ballycastle	BT37 5GW	33923231	jitweed@hotmail...	4	5	True
9	Diana	Ann	James	14/04/1985	6 Polyroad	Bushmills	BT83 5FE	40039237	dianajames@ya...	4	6	True
10	Lorraine	Hannah	Silvers	17/08/1999	48 union street	Portrush	BT84 2TR	40027364	loraines4silver...	1	7	False
11	Courtney	Lisa	Doherty	13/01/1994	11 jacobsLane	Gavagh	BT95 3RE	47326678	courtney2009...	1	8	True
12	Patrick	Samuel	Roberts	20/02/1973	92 Ballyway	Ballyboegey	BT05 5RF	30283736	patrick520@yahoo...	2	9	True
13	Matthew	Nathan	Jones	26/06/1975	45 tinkers street	Portstewart	BT82 7EQ	46335362	matthewjones...	3	2	False
14	Samantha	Alison	McFarlane	17/05/1982	231 Anderson A...	Ballymoney	BT00 3UN	45673623	sammy.mcfarlan...	2	4	True
15	Linda	Hayley	Spears	04/04/1996	999 Goldhill	Ballymena	BT71 9VR	37434794	lindahayley29...	3	5	False
16	Lucy	Susan	Dysart	28/07/1991	52 Colour avenue	Coleraine	BT52 1QA	50382627	tonyhoward@q...	4	6	True
17	Anthony	Ryan	Howard	12/03/1990	32 Hillview	Portrush	BT95 3VU	50947464	tonyhoward@q...	3	10	True
18	Cleo	Laura	Kane	23/02/2000	32 Hillview	Ballyboegey	BT59 2EC	3892374	cleopatra68@h...	4	2	True
19	Louise	Amanda	Doran	04/09/1994	39 Union street	Coleraine	BT51 3EH	75434934	louloodoran09...	3	5	True
20	Gerard	Josh	McDaid	25/04/1968	23 Chief Street	Bushmills	BT78 4YT	46637328	gerard300@ya...	1	7	False
21	Kayla	Brooke	Black	13/10/1978	91 butchroad	Ballymoney	BT94 2VU	53752852	kaylablack@ya...	2	6	False
22	Kristeen	Casey	Dickenson	14/11/1979	19 Screenhill ro...	BallyCastle	BT39 5GF	53278523	Kdickenson@Y...	2	10	False
23	Leanna	Skylar	Philips	01/01/1992	124 Strand Park	Gavagh	BT43 9KV	96854746	LeaSky942@Hot...	1	3	False
24	Duke	Kassy	Ewart	25/12/1998	220 Anderson A...	Portrush	BT59 4RM	57325632	DukeEwart@G...	1	3	False
25	Alan	Shantelle	Blue	21/08/2995	60 Tullyarton D...	BallyBoegey	BT79 0CJ	53275823	AlanBlue@Yah...	4	2	True
26	Reginald	Jessa	Stevenson	05/07/1982	14 Gelbe Park	Coleraine	BT46 2HD	13123934	rgs24@gmail.co...	1	9	False
27	Frederica	Trevelyan	Cobb	09/03/2000	84 Crescent Road	Coleraine	BT59 5GL	70329356	FredCobb@Hot...	3	4	True
0	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

2.4[A][1] Screenshot B

The screenshot shows the 'Student List' application window titled 'frmStudents'. The title bar features a logo of a guitar and the text 'The Mitchell School Of Music'. The main interface is titled 'Student List' in a large, stylized font. On the left, there is a form with various input fields for student information:

Student ID:	4
First Name:	Linda
Other Name(s):	Louise
Surname:	Simmons
Date Of Birth:	20/02/1998
Address:	14 Greenroad
Town:	Coleraine
Post Code:	BT51 2HE
Contact Number:	70359371
Email Address:	Linda854@gmail.com
Grade ID:	2
Instrument ID:	1
Tuition Fee Received:	True

On the right side, there is a control panel with the following buttons:

- Search for Student: Adam Ackles (dropdown menu)
- previous Student Record
- Next Student Record
- Add new Student
- Update Student Record
- Remove Student Record
- Return to Main Menu

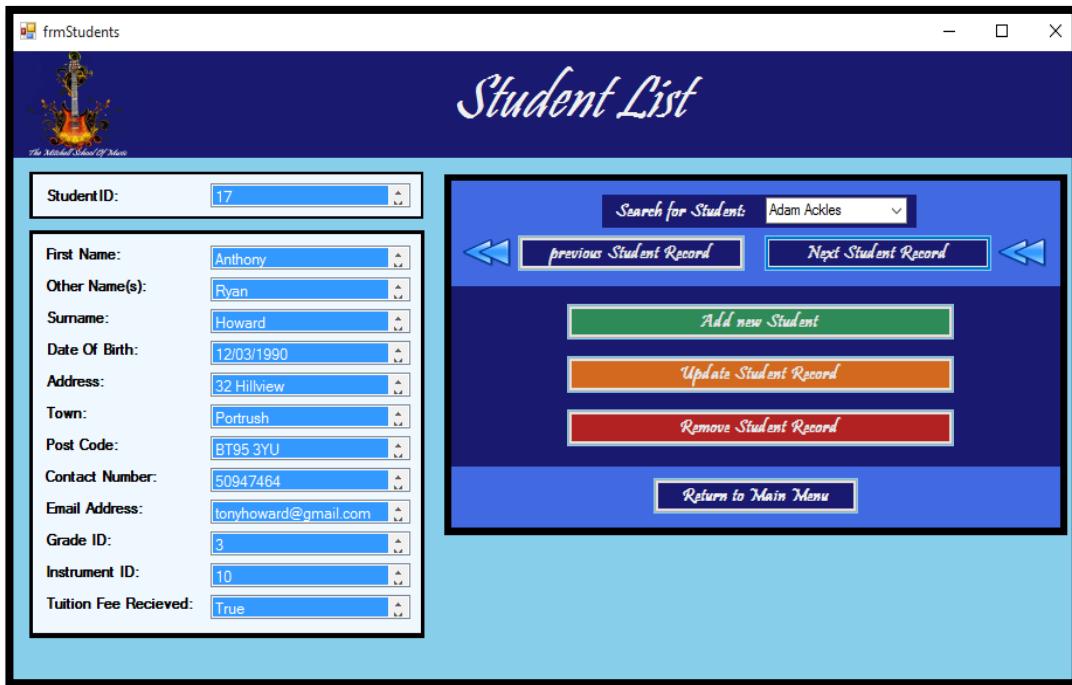
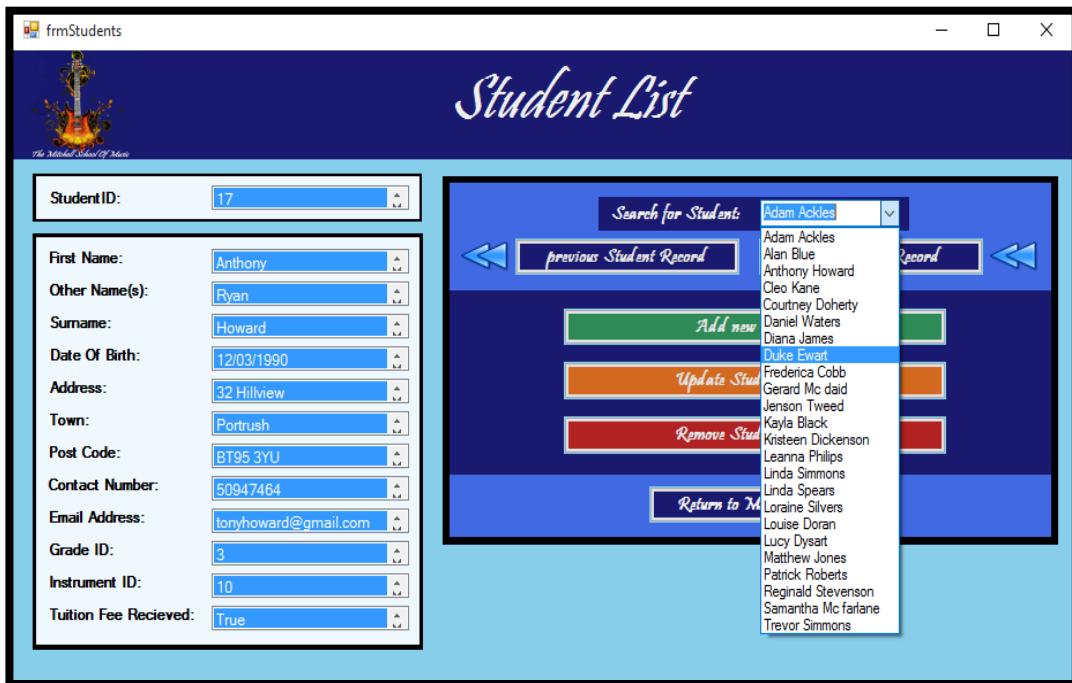
2.4[A][1] Screenshot C

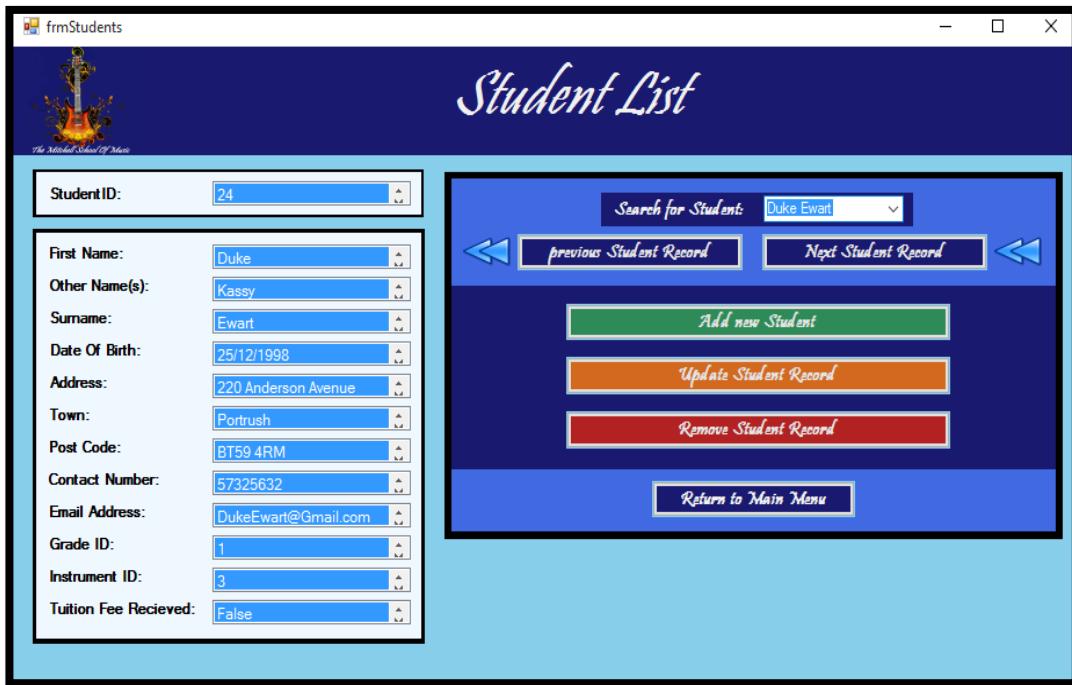
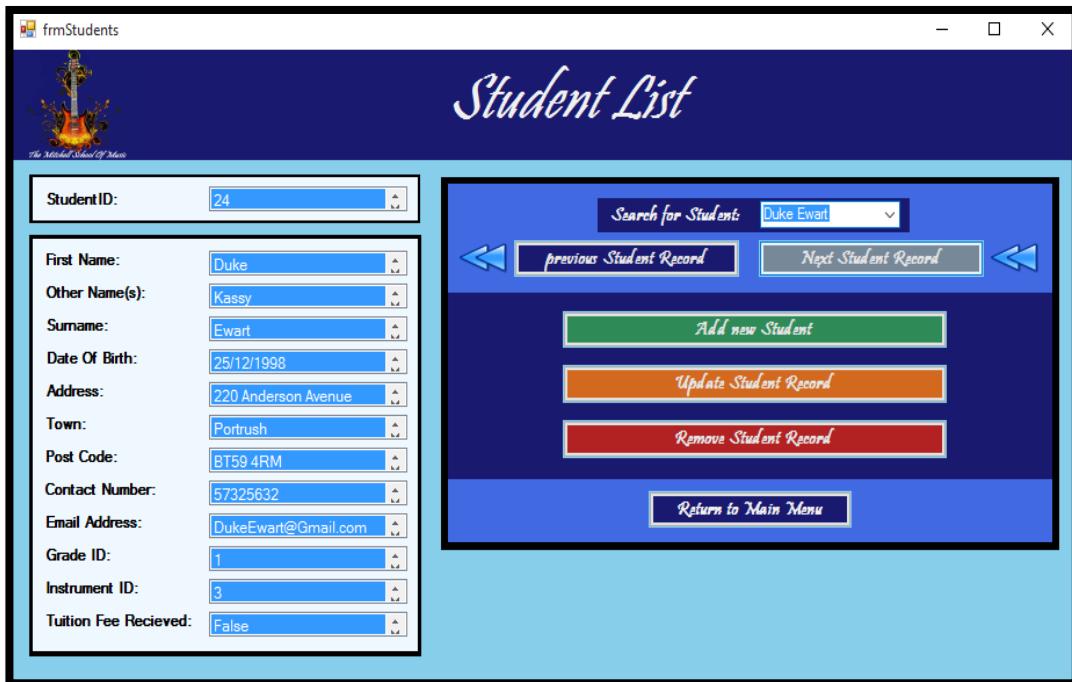
The screenshot shows the 'Student List' application window titled 'frmStudents'. The title bar features a logo of a guitar and the text 'The Mitchell School Of Music'. The main interface is titled 'Student List' in a large, stylized font. On the left, there is a form with various input fields for student information:

Student ID:	9
First Name:	Diana
Other Name(s):	Ann
Surname:	James
Date Of Birth:	14/04/1985
Address:	6 Polyroad
Town:	Bushmills
Post Code:	BT83 5FE
Contact Number:	40039237
Email Address:	dianajames@yahoo.com
Grade ID:	4
Instrument ID:	6
Tuition Fee Received:	True

On the right side, there is a control panel with the following buttons:

- Search for Student: Adam Ackles (dropdown menu)
- previous Student Record
- Next Student Record
- Add new Student
- Update Student Record
- Remove Student Record
- Return to Main Menu

2.4[A][1] Screenshot D**Process [B]) Record Navigation****2.4[B][1] – Student Search****2.4[B][1] Screenshot A**

2.4[B][1] Screenshot B**2.4[B][2] – Next Student****2.4[B][2] Screenshot A**

2.4[B][2] Screenshot B

The screenshot shows the 'Student List' application window titled 'frmStudents'. The title bar features a logo for 'The Musical School Of Music'. The main area is titled 'Student List' and contains a search bar with the text 'Duke Ewart'. Below the search bar are buttons for 'previous Student Record' and 'Next Student Record'. A large green button labeled 'Add new Student' is visible. On the left, there is a list of student details in a table format:

Student ID:	25
First Name:	Alan
Other Name(s):	Shantelle
Surname:	Blue
Date Of Birth:	21/08/2995
Address:	60 Tullyarton Drive
Town:	BallyBogey
Post Code:	BT99 0CJ
Contact Number:	53275823
Email Address:	AlanBlue@yahoo.com
Grade ID:	4
Instrument ID:	2
Tuition Fee Received:	True

2.4[B][3] – Previous Student**2.4[B][3] Screenshot A**

The screenshot shows the 'Student List' application window titled 'frmStudents'. The title bar features a logo for 'The Musical School Of Music'. The main area is titled 'Student List' and contains a search bar with the text 'Duke Ewart'. Below the search bar are buttons for 'previous Student Record' and 'Next Student Record'. A large blue button labeled 'Return to Main Menu' is visible. On the left, there is a list of student details in a table format, identical to Screenshot B:

Student ID:	25
First Name:	Alan
Other Name(s):	Shantelle
Surname:	Blue
Date Of Birth:	21/08/2995
Address:	60 Tullyarton Drive
Town:	BallyBogey
Post Code:	BT99 0CJ
Contact Number:	53275823
Email Address:	AlanBlue@yahoo.com
Grade ID:	4
Instrument ID:	2
Tuition Fee Received:	True

2.4[B][3] Screenshot B

The screenshot shows the frmStudents application window titled "Student List". The interface includes a logo for "The Musical School Of Music" at the top left. On the left side, there is a vertical stack of text input fields for student details: Student ID (24), First Name (Duke), Other Name(s) (Kassy), Surname (Ewart), Date Of Birth (25/12/1998), Address (220 Anderson Avenue), Town (Portrush), Post Code (BT59 4RM), Contact Number (57325632), Email Address (DukeEwart@Gmail.com), Grade ID (1), Instrument ID (3), and Tuition Fee Received (False). To the right of these fields is a central panel with a search bar labeled "Search for Student" containing "Duke Ewart". Below the search bar are four buttons: "previous Student Record" (left arrow), "Next Student Record" (right arrow), "Add new Student" (green button), "Update Student Record" (orange button), "Remove Student Record" (red button), and "Return to Main Menu" (blue button).

2.4[B][4] – First Student**2.4[B][1] Screenshot A**

The screenshot shows the frmStudents application window titled "Student List". The interface includes a logo for "The Musical School Of Music" at the top left. On the left side, there is a vertical stack of text input fields for student details: Student ID (27), First Name (Frederica), Other Name(s) (Trevelyan), Surname (Cobb), Date Of Birth (09/03/2000), Address (84 Crescent Road), Town (Coleraine), Post Code (BT59 5GL), Contact Number (70329356), Email Address (FredCobb@Hotmail.com), Grade ID (3), Instrument ID (4), and Tuition Fee Received (True). To the right of these fields is a central panel with a search bar labeled "Search for Student" containing "Adam Ackles". Below the search bar are four buttons: "previous Student Record" (left arrow), "Next Student Record" (right arrow), "Add new Student" (green button), "Update Student Record" (orange button), "Remove Student Record" (red button), and "Return to Main Menu" (blue button).

2.4[B][1] Screenshot B

The screenshot shows the 'Student List' application window titled 'frmStudents'. The title bar features a logo of a guitar and the text 'The Musical School Of Music'. The main area is titled 'Student List' in a large, stylized font. On the left, there is a form with various input fields for student information:

Student ID:	4
First Name:	Linda
Other Name(s):	Louise
Surname:	Simmons
Date Of Birth:	20/02/1998
Address:	14 Greenroad
Town:	Coleraine
Post Code:	BT51 2HE
Contact Number:	70359371
Email Address:	Linda854@gmail.com
Grade ID:	2
Instrument ID:	1
Tuition Fee Received:	True

On the right, there is a control panel with the following buttons:

- Search for Student: Duke Ewart (dropdown menu)
- previous Student Record
- Next Student Record
- Add new Student
- Update Student Record
- Remove Student Record
- Return to Main Menu

2.4[B][5] – Last Student**2.4[B][2] Screenshot A**

The screenshot shows the 'Student List' application window titled 'frmStudents'. The title bar features a logo of a guitar and the text 'The Musical School Of Music'. The main area is titled 'Student List' in a large, stylized font. On the left, there is a form with various input fields for student information:

Student ID:	27
First Name:	Frederica
Other Name(s):	Trevelyan
Surname:	Cobb
Date Of Birth:	09/03/2000
Address:	84 Crescent Road
Town:	Coleraine
Post Code:	BT59 5GL
Contact Number:	70329356
Email Address:	FredCobb@Hotmail.com
Grade ID:	3
Instrument ID:	4
Tuition Fee Received:	True

On the right, there is a control panel with the following buttons:

- Search for Student: Adam Ackles (dropdown menu)
- previous Student Record
- Next Student Record
- Add new Student
- Update Student Record
- Remove Student Record
- Return to Main Menu

2.4[B][2] Screenshot B

The screenshot shows the frmStudents application window titled "Student List". On the left, there is a form with various dropdown menus for student information: First Name (Frederica), Other Name(s) (Trevelyan), Surname (Cobb), Date Of Birth (09/03/2000), Address (84 Crescent Road), Town (Coleraine), Post Code (BT59 5GL), Contact Number (70329356), Email Address (FredCobb@Hotmail.com), Grade ID (3), Instrument ID (4), and Tuition Fee Received (True). On the right, there is a control panel with buttons for "Search for Student" (set to Duke Ewart), "previous Student Record" and "Next Student Record" (with double-headed arrows), and three main action buttons: "Add new Student" (green), "Update Student Record" (orange), and "Remove Student Record" (red). Below these is a "Return to Main Menu" button.

Process [C]) Form Navigation**2.4[C][1] – Add New Student Record****2.4[C][1] Screenshot A**

This screenshot is identical to Screenshot B, showing the frmStudents application window titled "Student List". It displays the same form on the left for entering student details and the same control panel on the right with the "Add new Student" button highlighted in grey, indicating it is the active or selected option.

2.4[C][1] Screenshot B

The screenshot shows a Windows application window titled "frmAddField" with a dark blue header bar. The title bar contains the text "Add Student". In the top left corner, there is a logo for "The National School Of Music" featuring a stylized guitar. Below the title bar, there are two buttons: "Add New Student" on the left and "Return to Students Table" on the right.

The main area of the window contains several text input fields and dropdown menus:

- First Name: [Text Box]
- Other Name(s): [Text Box]
- Surname: [Text Box]
- Date Of Birth: [Text Box] with a small calendar icon to its right.
- Address: [Text Box]
- Town: [Text Box]
- PostCode: [Text Box]
- Contact Number: [Text Box]
- Email Address: [Text Box]
- GradeID: [Dropdown Menu] showing value "1"
- InstrumentID: [Dropdown Menu] showing value "1"
- Tuition Fee Received: [Check Box]

2.4[C][2] – Update Student Record**2.4[C][2] Screenshot A**

The screenshot shows a Windows application window titled "frmStudents" with a dark blue header bar. The title bar contains the text "Student List". In the top left corner, there is a logo for "The National School Of Music" featuring a stylized guitar. Below the title bar, there is a search bar labeled "Search for Student: Adam Ackles" with a dropdown arrow, flanked by "previous Student Record" and "Next Student Record" buttons.

The main area of the window contains several text input fields and dropdown menus, identical to the ones in the previous screenshot:

- StudentID: [Text Box] showing value "4"
- First Name: [Text Box] showing value "Linda"
- Other Name(s): [Text Box] showing value "Louise"
- Surname: [Text Box] showing value "Simmons"
- Date Of Birth: [Text Box] showing value "20/02/1998"
- Address: [Text Box] showing value "14 Greenroad"
- Town: [Text Box] showing value "Coleraine"
- Post Code: [Text Box] showing value "BT51 2HE"
- Contact Number: [Text Box] showing value "70359371"
- Email Address: [Text Box] showing value "Linda854@gmail.com"
- Grade ID: [Dropdown Menu] showing value "2"
- Instrument ID: [Dropdown Menu] showing value "1"
- Tuition Fee Received: [Text Box] showing value "True"

Below the input fields, there are three buttons:

- Add new Student (Green button)
- Update Student Record (Grey button)
- Remove Student Record (Red button)

At the bottom of the window is a "Return to Main Menu" button.

2.4[C][2] Screenshot B

The screenshot shows a Windows application window titled "frmAddField". The title bar features a logo of a guitar and the text "The Selected School Of Music". The main window is titled "Add Student" and contains a "Add New Student" button and a "Return to Students Table" button. The form includes fields for Student ID (4), First Name (Linda), Other Name(s) (Louise), Surname (empty), Date Of Birth (20/02/1998), Address (14 Greenroad), Town (Coleraine), PostCode (BT51 2HE), Contact Number (70359371), Email Address (Linda854@gmail.com), GradeID (1), InstrumentID (1), and Tuition Fee Received (checkbox checked). There is also a small thumbnail image of a person next to the address field.

2.4[C][2][Corrective Action] – Update Student Record**2.4[C][Corrective Action] Screenshot A**

The screenshot shows the code editor for the frmAddField.cs file. The code defines a class frmAddField with various properties and methods. It includes lists for students, teachers, lesson bundles, and scheduled lessons. It also handles database connections and message boxes. The constructor initializes components and sets up event handlers for the Load event.

```

GlobalVariables.cs  frmAddField.cs*  frmAddField.cs [Design]*  frmStudents.cs  frmStudents.cs [Design]  frmTeachers.cs  frmTeachers.cs [Design]
frmTeachers.cs [Design]  frmMain.cs  frmPrivateTuition.cs [Design]  frmPrivateTuition.cs  frmSplash.cs  frmInstrument.cs [Design]  frmInstrument.cs [Design]
frmGrade.cs  frmRoom.cs  frmScheduleTable.cs  frmLessonBundle.cs [Design]  frmLessonBundle.cs
dbo.LessonsPurchased [Data]  dbo.LessonsPurchased [Design]  frmSplash.frmAddField  frmAddField()
Mitchell_School_Of_Music  frmAddField()  frmAddField()

List<string> Students_FullName_AND_StudentID = new List<string>();
List<int> Teachers_TeacherID = new List<int>();
List<string> Teachers_FirstName = new List<string>();
List<string> Teachers_Surname = new List<string>();
List<string> Teachers_FullName_AND_TeacherID = new List<string>();

List<int> LessonBundle_BundleID = new List<int>();
List<string> LessonBundle_Name = new List<string>();
List<string> LessonBundle_Name_AND_BundleID = new List<string>();

List<string> Timelist = new List<string>();

List<int> ScheduledLessons_PurchaseID = new List<int>();
List<int> purchasedLessons_IDs = new List<int>();

int Desired_StudentID;

List<string> ScheduleWeeks = new List<string>();

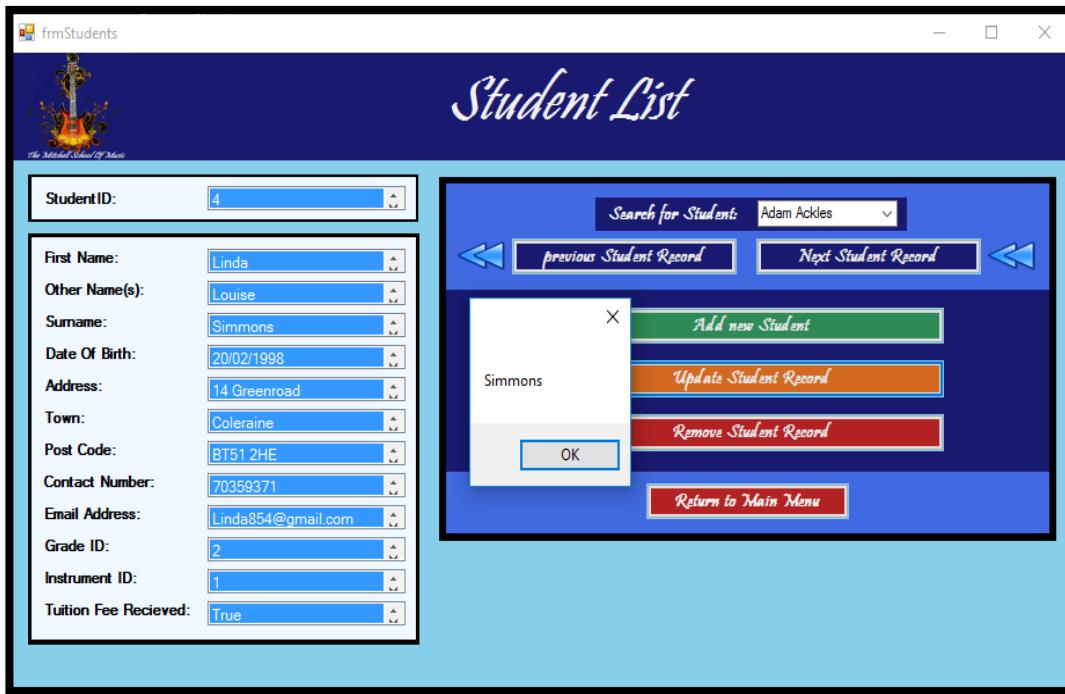
List<string> instruments = new List<string>();

public frmAddField()
{
    InitializeComponent();
    connectionString = ConfigurationManager.ConnectionStrings["frmSplash.Properties.Settings.PrivateTuitionConnectionString"].ConnectionString;
    MessageBox.Show(GlobalVariables.Student_Surname);
}

private void frmAddField_Load(object sender, EventArgs e)
{
    Hide_Addition1();
    Hide_Error();
    Store_StudentInformation();
    Store_TeacherInformation();
    Store_BundleInformation();
    Store_ScheduleInformation();
    Store_PurchaseInformation();

    TotalInstruments = 20;
    TotalGradeRange = 88;
    TotalBundleRange = 100;
}

```

2.4[C][Corrective Action] Screenshot B**2.4[C][Corrective Action] Screenshot C**

The screenshot shows a Microsoft Visual Studio code editor displaying the file "frmSplash.frmAddField.cs". The code is written in C# and contains logic for handling different forms based on previous form names and purpose. It includes methods for hiding various controls like picture boxes and grids, and for showing calendar controls. It also includes event handlers for button clicks and dropdown selection changes, such as "btmDataGrid_Click" and "cbFieldInfo1_SelectedIndexChanged". The code uses GlobalVariables and frmSplash classes to manage the state of the application.

```

GlobalVariables.cs  frmAddField.cs*  frmAddField.cs [Design]*  frmStudents.cs  frmStudents.cs [Design]  frmTeachers.cs
frmTeachers.cs [Design]  frmMain.cs  frmPrivateTuition.cs [Design]  frmPrivateTuition.cs  frmSplash.cs  frmInstrument.cs [Design]
frmGrade.cs  frmRoom.cs  frmScheduleTable.cs
dbo.LessonsPurchased [Data]  dbo.LessonsPurchased [Design]  frmInstrument.cs  frmLessonBundle.cs [Design]  frmLessonBundle.cs
frmSplash.frmAddField
private void pictureBox1_Hide();
DataGrid_PurchasedLessons.Hide();
Calender1.Hide();

if (GlobalVariables.PreviousForm == "ScheduledLessonTable" || GlobalVariables.PreviousForm == "Calender")
{
    btmCalendar.Show();
    btmDataGrid.Show();
    btnTimeSelection.Show();

    if (GlobalVariables.Purpose == "Update")
    {
        btmCalendar2.Show();
    }
    else if (GlobalVariables.PreviousForm == "StudentTable")
    {
        btmCalendar.Show();
    }
}

Inherits
private void btmDataGrid_Click(object sender, EventArgs e)
{
}

Inherits
private void cbFieldInfo1_SelectedIndexChanged(object sender, EventArgs e)
{
    Student_Search_Breakdown();
    tbFieldInfo4.Text = null;
}

Inherits
public void Student_Search_Breakdown()
{
}

Inherits
private void DataGrid_PurchasedLessons_SelectionChanged(object sender, EventArgs e)
{
    if (DataGrid_PurchasedLessons.SelectedCells.Count > 0)
    {
        int selectedRowIndex = DataGrid_PurchasedLessons.SelectedCells[0].RowIndex;
    }
}

```

2.4[C][Corrective Action] Screenshot D

```

using System;
using System.Windows.Forms;

namespace Mitchell_School_of_Music
{
    public partial class frmAddField : Form
    {
        // Other code and declarations omitted for brevity
    }
}

```

2.4[C][Corrective Action] Screenshot E

Student ID:	4
First Name:	Linda
Other Name(s):	Louise
Surname:	Simmons
Date Of Birth:	20/02/1998
Address:	14 Greenroad
Town:	Coleraine
PostCode:	BT51 2HE
Contact Number:	70359371
Email Address:	Linda854@gmail.com
GradeID:	1
InstrumentID:	1
Tuition Fee Received:	<input checked="" type="checkbox"/>

2.4[C][3] – Delete Student Record [User Logged In]**2.4[C][3] Screenshot A**

The screenshot shows a Windows application window titled "frmStudents". The title bar has a logo of a guitar and the text "The Mitchell School Of Music". The main title of the window is "Student List". On the left, there is a form with various input fields for student information: Student ID (4), First Name (Linda), Other Name(s) (Louise), Surname (Simmons), Date Of Birth (20/02/1998), Address (14 Greenroad), Town (Coleraine), Post Code (BT51 2HE), Contact Number (70359371), Email Address (Linda854@gmail.com), Grade ID (2), Instrument ID (1), and Tuition Fee Received (True). On the right, there is a control panel with buttons: "Search for Student" (set to Adam Ackles), "previous Student Record" (disabled), "Next Student Record" (disabled), "Add new Student" (green button), "Update Student Record" (orange button), "Remove Student Record" (grey button), and "Return to Main Menu" (blue button).

2.4[C][3] Screenshot B

The screenshot shows a Windows application window titled "frmUserLogin". The title bar has a logo of a guitar and the text "The Mitchell School Of Music". The main title of the window is "Admin Login". It features two text input fields: "User Name:" and "Password:", both currently empty. At the bottom, there are two buttons: a red "Return" button on the left and a green "Log In" button on the right.

2.4[C][4] – Delete Student Record [No User Logged In]**2.4[C][4] Screenshot A**

frmStudents

Student List



Search for Student: Adam Ackles

previous Student Record Next Student Record

Add new Student Update Student Record Remove Student Record

Return to Main Menu

Student ID:	4
First Name:	Linda
Other Name(s):	Louise
Surname:	Simmons
Date Of Birth:	20/02/1998
Address:	14 Greenroad
Town:	Coleraine
Post Code:	BT51 2HE
Contact Number:	70359371
Email Address:	Linda854@gmail.com
Grade ID:	2
Instrument ID:	1
Tuition Fee Received:	True

2.4[C][4] Screenshot B

frmStudents

Student List



Search for Student: Adam Ackles

previous Student Record Next Student Record

Add new Student Update Student Record Remove Student Record

Return to Main Menu

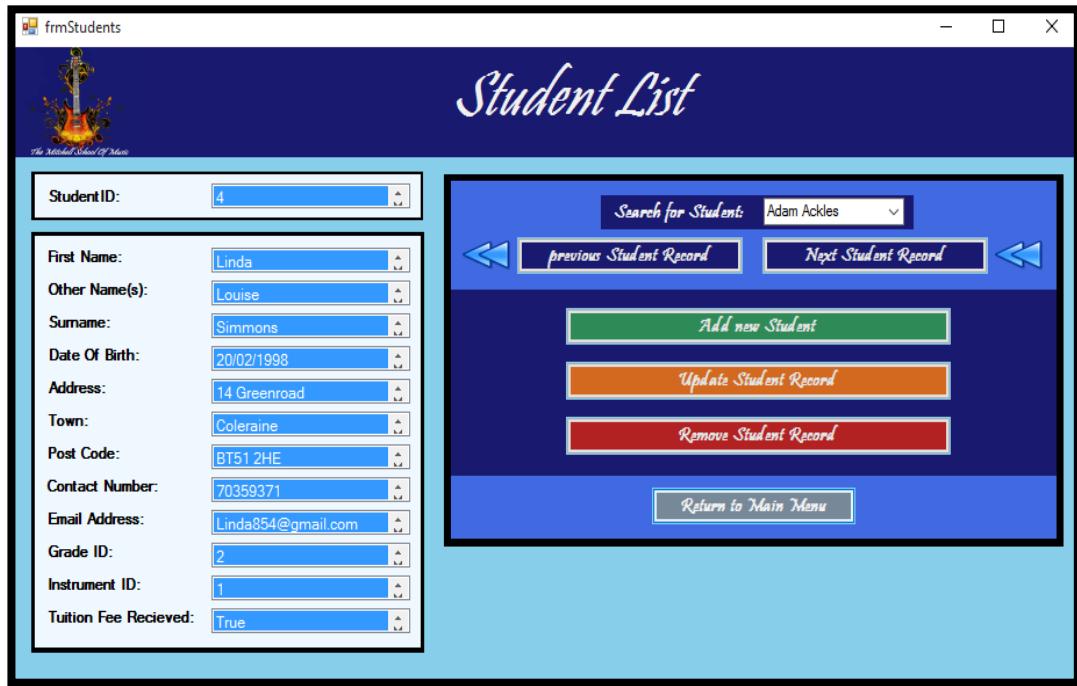
Student ID:	4
First Name:	Linda
Other Name(s):	Louise
Surname:	Simmons
Date Of Birth:	20/02/1998
Address:	14 Greenroad
Town:	Coleraine
Post Code:	BT51 2HE
Contact Number:	70359371
Email Address:	Linda854@gmail.com
Grade ID:	2
Instrument ID:	1
Tuition Fee Received:	True

frmConfirmation

Delete Confirmation

Student ID: 4 Student Name: Linda Simmons

Cancel **Confirm**

2.4[C][5] – Return To Main Menu**2.4[C][5] Screenshot A****2.4[C][5] Screenshot B**

2.5) Teachers Form*Process [A] Information Display****2.5[A][1] – Teacher Information******2.5[A][1] Screenshot A***

	TeacherID	First_Name	Surname	Address	Town	PostCode	Email_Address	Contact_Num...	Specialisation	RoomID
D	1	Sandra	Smith	362 Atlantic Ro...	Coleraine	BT32 3RE	ssmith362@gm...	84933740	String	1
2	Carey	Gamble		84 Coleraine Ro...	Garvagh	BT84 8IE	careygamble82...	47119403	Wood Wind	2
3	Donna	Shaw		205 Strand Roa...	Ballymoney	BT04 3RD	donashaw@gm...	49573628	Keyboard	3
4	Nathan	Jones		320 Church Stre...	Castlerock	BT56 3QB	njones542@hot...	47118394	Precussion	4
5	Garry	Gibson		102 Portrush Ro...	Coleraine	BT29 9MV	grgibson920@g...	47118574	Brass	5
6	Jamie	Dome		162 Strand Roa...	Ballymoney	BT84 5BP	jdome@gmail....	48912647	Vocal	6
O	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

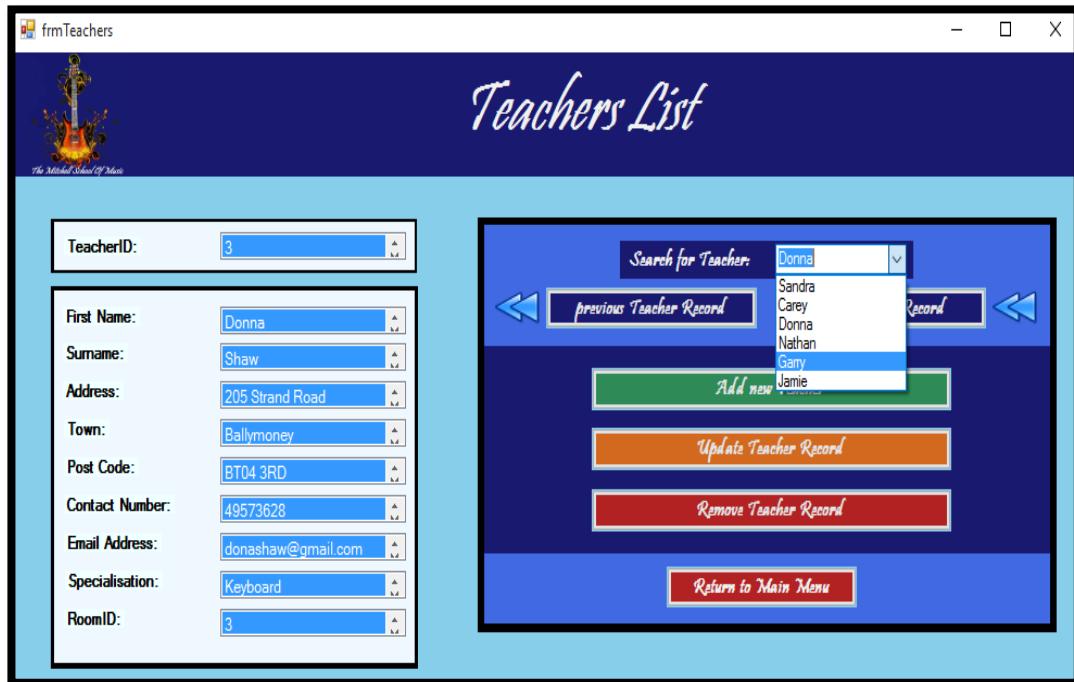
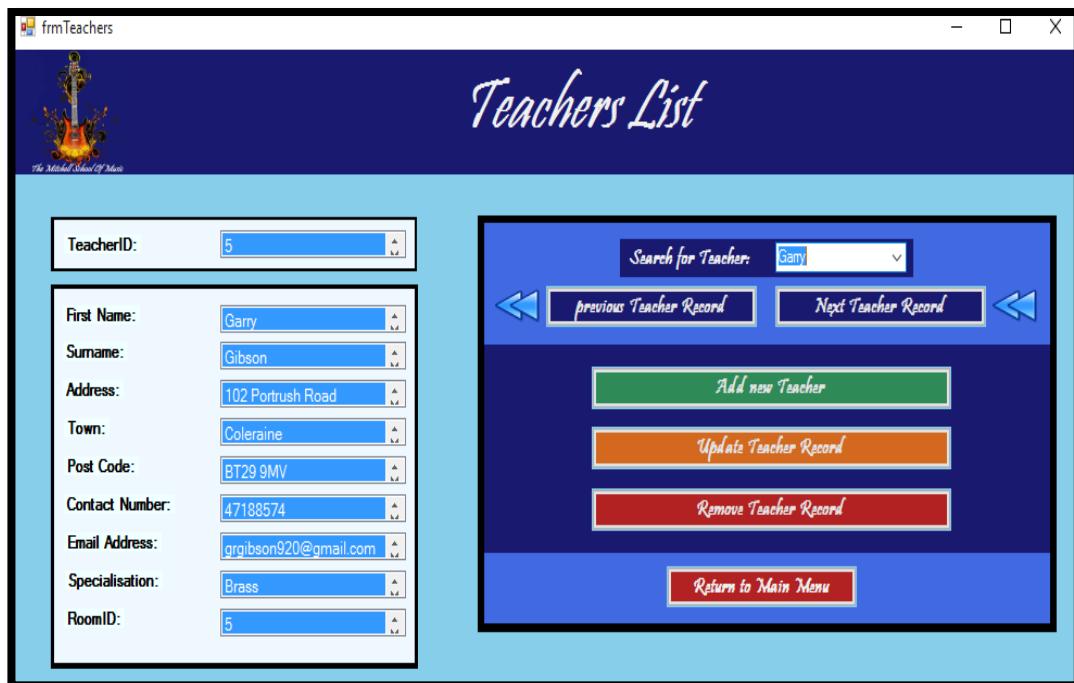
2.5[A][1] Screenshot B

2.5[A][1] Screenshot C

The screenshot shows the frmTeachers application window titled "Teachers List". The main area displays a teacher record for "Carey". The record includes fields for TeacherID (2), First Name (Carey), Surname (Gamble), Address (84 Coleraine Road), Town (Garvagh), Post Code (BT84 8IE), Contact Number (47119403), Email Address (careygamble82@yahoo.co.uk), Specialisation (Wood Wind), and RoomID (2). On the right side, there is a control panel with a search bar set to "Carey", and buttons for "previous Teacher Record", "Next Teacher Record", "Add new Teacher" (green), "Update Teacher Record" (orange), "Remove Teacher Record" (red), and "Return to Main Menu".

2.5[A][1] Screenshot D

The screenshot shows the frmTeachers application window titled "Teachers List". The main area displays a teacher record for "Donna". The record includes fields for TeacherID (3), First Name (Donna), Surname (Shaw), Address (205 Strand Road), Town (Ballymoney), Post Code (BT04 3RD), Contact Number (49573628), Email Address (donashaw@gmail.com), Specialisation (Keyboard), and RoomID (3). On the right side, there is a control panel with a search bar set to "Donna", and buttons for "previous Teacher Record", "Next Teacher Record", "Add new Teacher" (green), "Update Teacher Record" (orange), "Remove Teacher Record" (red), and "Return to Main Menu".

*Process [B]) Record Navigation***2.5[B][1] – Teacher Search****2.5[B][1] Screenshot A****2.5[B][1] Screenshot B**

2.5[B][2] – Next Teacher**2.5[B][2] Screenshot A**

The screenshot shows the 'Teachers List' application window. On the left, there is a form with fields for TeacherID (5), First Name (Garry), Surname (Gibson), Address (102 Portrush Road), Town (Coleraine), Post Code (BT29 9MV), Contact Number (47188574), Email Address (grgibson920@gmail.com), Specialisation (Brass), and RoomID (5). On the right, a central panel displays a search bar with 'Search for Teacher: Garry' and navigation buttons for 'previous Teacher Record' and 'Next Teacher Record'. Below these are three buttons: 'Add new Teacher' (green), 'Update Teacher Record' (orange), and 'Remove Teacher Record' (red). At the bottom is a 'Return to Main Menu' button.

2.5[B][2] Screenshot B

The screenshot shows the 'Teachers List' application window. On the left, there is a form with fields for TeacherID (6), First Name (Jamie), Surname (Dome), Address (162 Strand Road), Town (Ballymoney), Post Code (BT84 5BP), Contact Number (48912647), Email Address (jdome@gmail.co.uk), Specialisation (Vocal), and RoomID (6). On the right, a central panel displays a search bar with 'Search for Teacher: Jamie' and navigation buttons for 'previous Teacher Record' and 'Next Teacher Record'. Below these are three buttons: 'Add new Teacher' (green), 'Update Teacher Record' (orange), and 'Remove Teacher Record' (red). At the bottom is a 'Return to Main Menu' button.

2.5[B][3] – Previous Teacher**2.5[B][3] Screenshot A**

The screenshot shows a Windows application window titled "frmTeachers". The title bar features a logo of a guitar and the text "The Musical School Of Music". The main title "Teachers List" is displayed prominently at the top center. On the left, there is a form with fields for "TeacherID" (set to 6), "First Name" (Jamie), "Surname" (Dome), "Address" (162 Strand Road), "Town" (Ballymoney), "Post Code" (BT84 5BP), "Contact Number" (48912647), "Email Address" (jdome@gmail.co.uk), "Specialisation" (Vocal), and "RoomID" (6). On the right, a control panel includes a "Search for Teacher" dropdown set to "Jamie", "previous Teacher Record" and "Next Teacher Record" buttons, and three action buttons: "Add new Teacher" (green), "Update Teacher Record" (orange), and "Remove Teacher Record" (red). A "Return to Main Menu" button is also present.

2.5[B][3] Screenshot B

The screenshot shows the same Windows application window as in Screenshot A. The title bar and logo are identical. The main title "Teachers List" is at the top center. The left side displays a form with fields for "TeacherID" (set to 5), "First Name" (Garry), "Surname" (Gibson), "Address" (102 Portrush Road), "Town" (Coleraine), "Post Code" (BT29 9MV), "Contact Number" (47188574), "Email Address" (grgibson920@gmail.com), "Specialisation" (Brass), and "RoomID" (5). The right side features a "Search for Teacher" dropdown set to "Garry", "previous Teacher Record" and "Next Teacher Record" buttons, and three action buttons: "Add new Teacher" (green), "Update Teacher Record" (orange), and "Remove Teacher Record" (red). A "Return to Main Menu" button is also present.

2.5[B][4] – First Teacher**2.5[B][4] Screenshot A**

The screenshot shows the frmTeachers application window titled "Teachers List". On the left, there is a form with fields for TeacherID (6), First Name (Jamie), Surname (Dome), Address (162 Strand Road), Town (Ballymoney), Post Code (BT84 5BP), Contact Number (48912647), Email Address (jdome@gmail.co.uk), Specialisation (Vocal), and RoomID (6). On the right, there is a control panel with a search field ("Search for Teacher: Jamie"), navigation buttons ("previous Teacher Record" and "Next Teacher Record"), and three action buttons: "Add new Teacher" (green), "Update Teacher Record" (orange), and "Remove Teacher Record" (red). A "Return to Main Menu" button is also present.

2.5[B][4] Screenshot B

The screenshot shows the frmTeachers application window titled "Teachers List". On the left, there is a form with fields for TeacherID (1), First Name (Sandra), Surname (Smith), Address (362 Atlantic Road), Town (Coleraine), Post Code (BT32 3RE), Contact Number (84933740), Email Address (ssmith362@gmail.co.uk), Specialisation (String), and RoomID (1). On the right, there is a control panel with a search field ("Search for Teacher: Sandra"), navigation buttons ("previous Teacher Record" and "Next Teacher Record"), and three action buttons: "Add new Teacher" (green), "Update Teacher Record" (orange), and "Remove Teacher Record" (red). A "Return to Main Menu" button is also present.

2.5[B][5] – Last Teacher**2.5[B][5] Screenshot A**

The screenshot shows the 'Teachers List' application window. On the left, there is a form for entering teacher details. On the right, there is a control panel with search and navigation buttons.

TeacherID: 1

First Name: Sandra

Surname: Smith

Address: 362 Atlantic Road

Town: Coleraine

Post Code: BT32 3RE

Contact Number: 84933740

Email Address: ssmith362@gmail.co.uk

Specialisation: String

RoomID: 1

Search for Teacher: Sandra

previous Teacher Record | **Next Teacher Record**

Add new Teacher

Update Teacher Record

Remove Teacher Record

Return to Main Menu

2.5[B][5] Screenshot B

The screenshot shows the 'Teachers List' application window. On the left, there is a form for entering teacher details. On the right, there is a control panel with search and navigation buttons.

TeacherID: 6

First Name: Jamie

Surname: Dome

Address: 162 Strand Road

Town: Ballymoney

Post Code: BT84 5BP

Contact Number: 48912647

Email Address: jdome@gmail.co.uk

Specialisation: Vocal

RoomID: 6

Search for Teacher: Jamie

previous Teacher Record | **Next Teacher Record**

Add new Teacher

Update Teacher Record

Remove Teacher Record

Return to Main Menu

*Process [C]) Form Navigation**2.5[C][1] – Add New Teacher Record**2.5[C][1] Screenshot A*

frmTeachers

The screenshot shows a Windows application window titled "frmTeachers". The title bar includes the application name and standard window controls. The main area has a dark blue header with the text "Teachers List" in a large, white, cursive font. Below the header is a light blue section containing a search interface. It features a text input field labeled "Search for Teacher:" with the value "Gary", flanked by "previous Teacher Record" and "Next Teacher Record" buttons. Below this are three buttons: "Add new Teacher" (grey), "Update Teacher Record" (orange), and "Remove Teacher Record" (red). At the bottom is a "Return to Main Menu" button. To the left of the search interface is a vertical stack of text input fields for teacher details, each with a label and a dropdown arrow. The fields are: TeacherID (5), First Name (Gary), Surname (Gibson), Address (102 Portrush Road), Town (Coleraine), Post Code (BT29 9MV), Contact Number (47188574), Email Address (grgibson920@gmail.com), Specialisation (Brass), and RoomID (5).

2.5[C][1] Screenshot B

frmAddField

The screenshot shows a Windows application window titled "frmAddField". The title bar includes the application name and standard window controls. The main area has a dark blue header with the text "Add Teacher" in a large, white, cursive font. Below the header is a light blue section containing two buttons: "Add New Teacher" (white) and "Return to Teacher Table" (red). The lower part of the window contains a form with text input fields for teacher details, each with a label and a dropdown arrow. The fields are: First Name, Surname, Address, Town, PostCode, Email Address, Contact Number, Specialisation (Brass), and RoomID (1).

2.5[C][2] – Update Teacher Record***2.5[C][2] Screenshot A***

frmTeachers

Teachers List

TeacherID: 1

First Name:	Sandra
Surname:	Smith
Address:	362 Atlantic Road
Town:	Coleraine
Post Code:	BT32 3RE
Contact Number:	84933740
Email Address:	ssmith362@gmail.co.uk
Specialisation:	String
RoomID:	1

Search for Teacher: Sandra

 << previous Teacher Record Next Teacher Record >>

 Add new Teacher (Green)

 Update Teacher Record (Grey)

 Remove Teacher Record (Red)

 Return to Main Menu

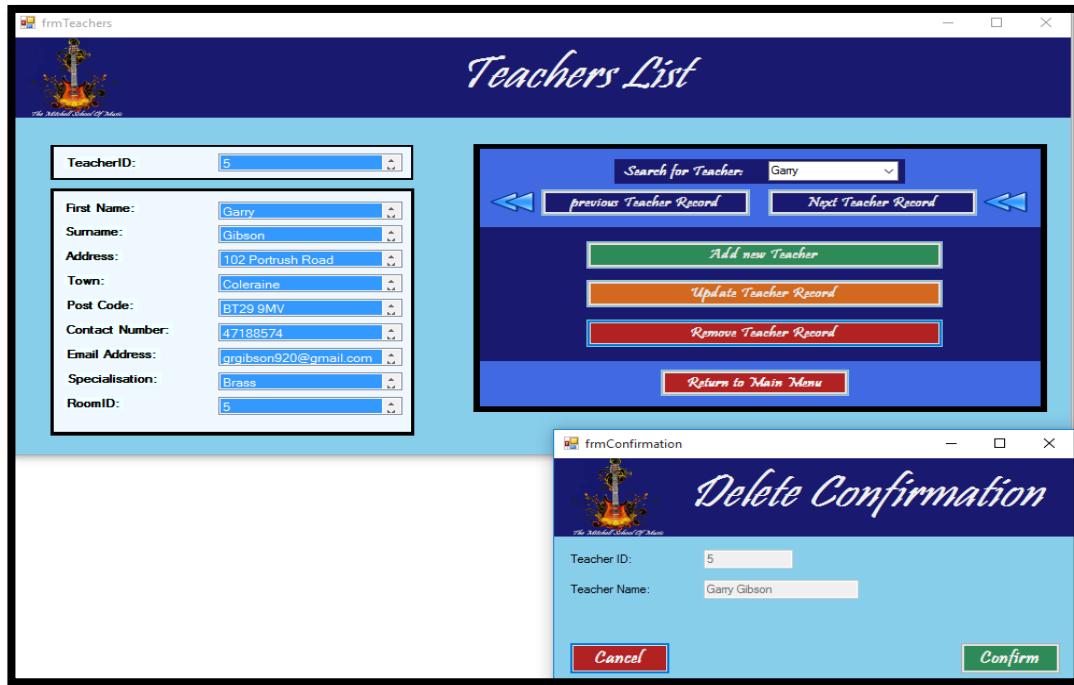
2.5[C][2] Screenshot B

frmAddField

Add Teacher

Add New Teacher Return to Teacher Table

Teacher ID	1
First Name:	Sandra
Surname:	Smith
Address:	362 Atlantic Road
Town:	Coleraine
PostCode:	BT32 3RE
Email Address:	ssmith362@gmail.co.uk
Contact Number:	84933740
Specialisation:	Brass
RoomID:	1

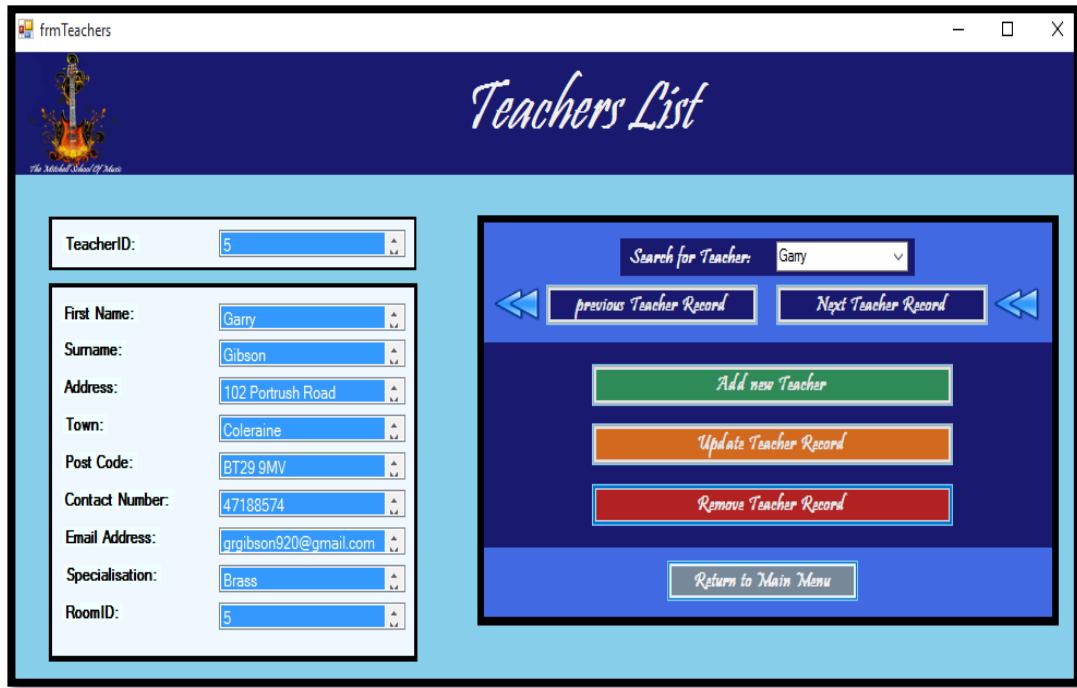
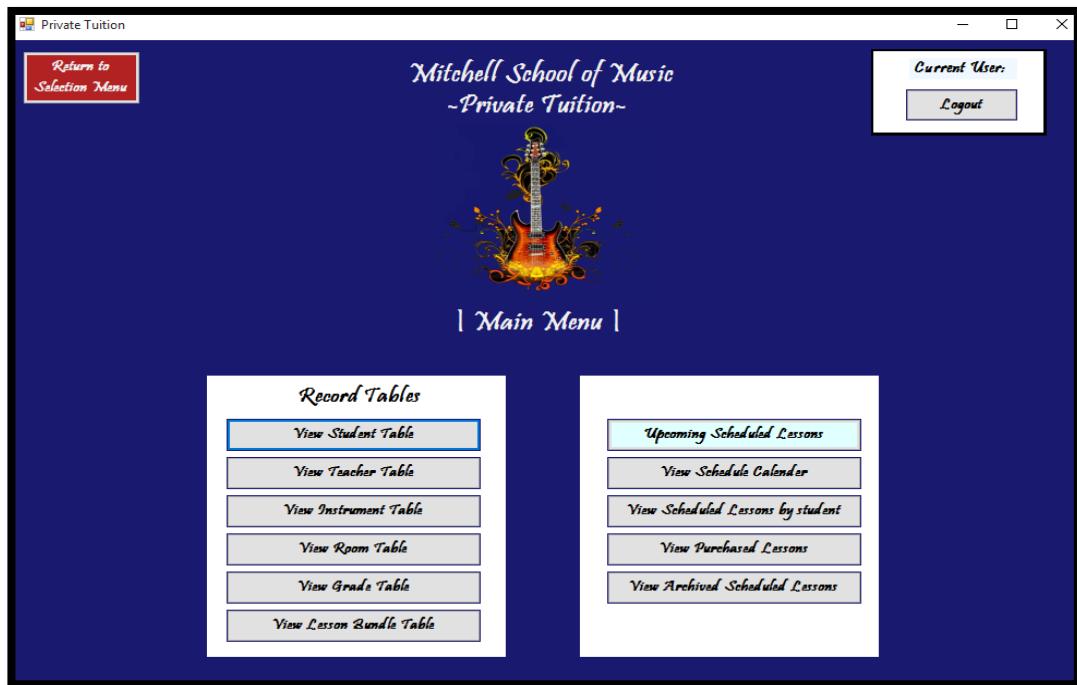
2.5[C][3] – Delete Teacher Record [User Logged In]**2.5[C][3] Screenshot A****2.5[C][3] Screenshot B**

2.5[C][4] – Delete Teacher Record [No User Logged In]**2.5[C][4] Screenshot A**

The screenshot shows a Windows application window titled "frmTeachers". The title bar features a logo of a guitar and the text "The Mitchell School Of Music". The main title of the window is "Teachers List". On the left, there is a form with fields for "TeacherID" (set to 5), "First Name" (Gary), "Surname" (Gibson), "Address" (102 Portrush Road), "Town" (Coleraine), "Post Code" (BT29 9MV), "Contact Number" (47188574), "Email Address" (grgibson92@gmail.com), "Specialisation" (Brass), and "RoomID" (5). On the right, there is a control panel with a search field containing "Gary", "previous Teacher Record" and "Next Teacher Record" buttons, and three buttons: "Add new Teacher" (green), "Update Teacher Record" (orange), and "Remove Teacher Record" (grey). A "Return to Main Menu" button is at the bottom.

2.5[C][4] Screenshot B

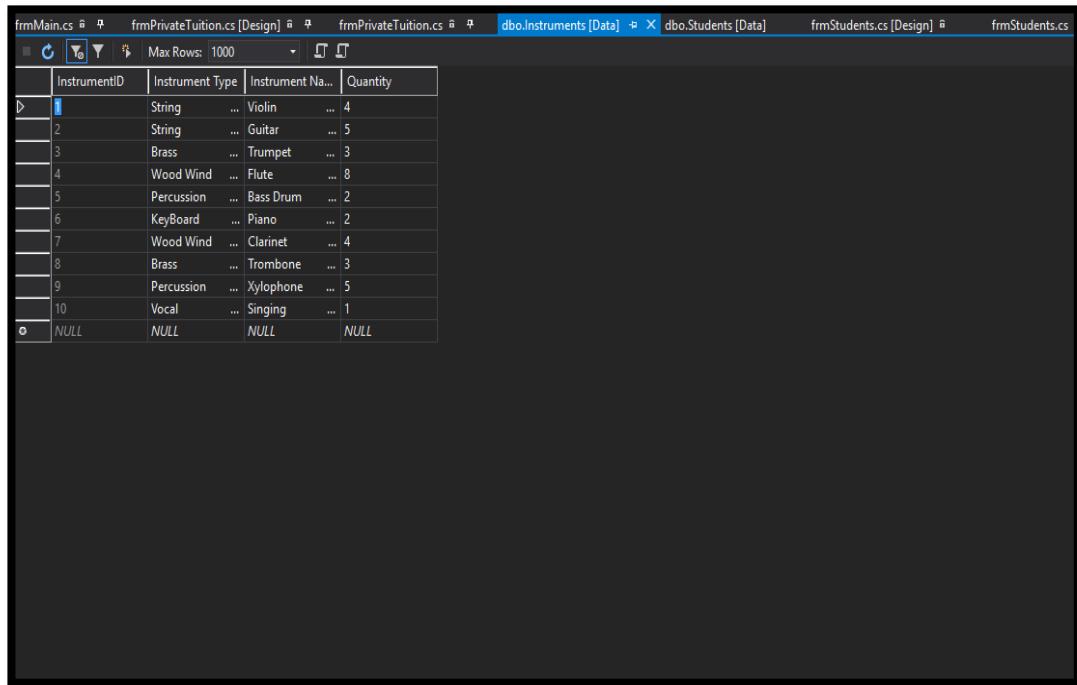
The screenshot shows a Windows application window titled "frmUserLogin". The title bar features a logo of a guitar and the text "The Mitchell School Of Music". The main title of the window is "Admin Login". It contains two text input fields: "User Name:" and "Password:", both currently empty. At the bottom, there are two buttons: a red "Return" button on the left and a green "Log In" button on the right.

2.5[C][5] – Return To Main Menu**2.5[C][5] Screenshot A****2.5[C][5] Screenshot B**

2.6) Instrument Form
Process [A] Information Display

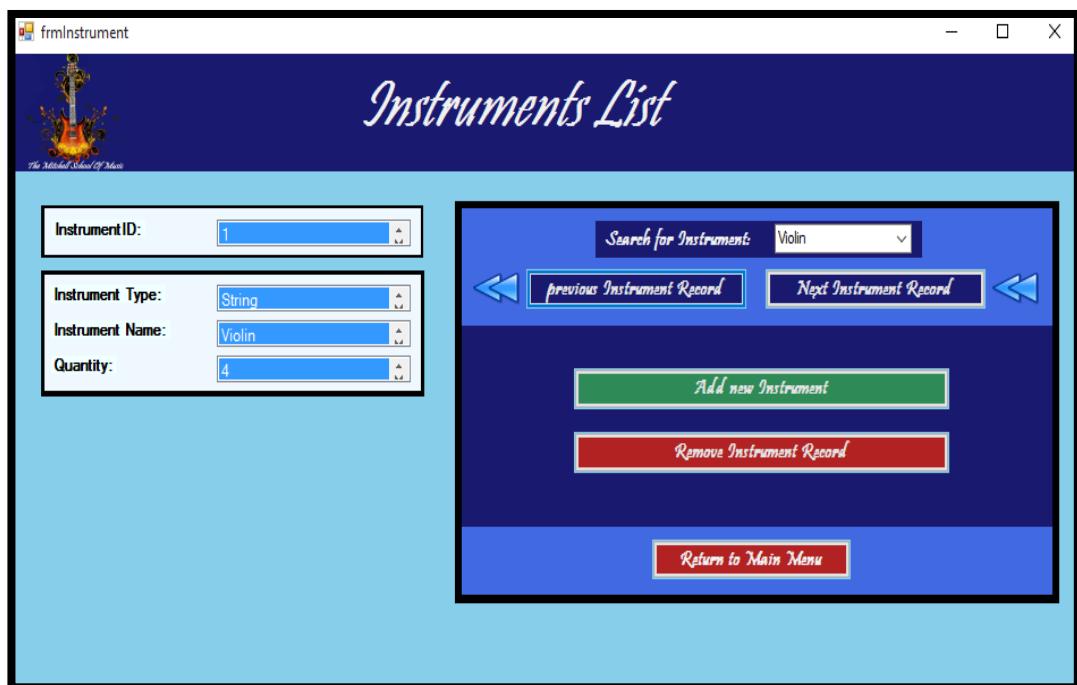
2.6[A][1] – Instrument Information

2.6[A][1] Screenshot A



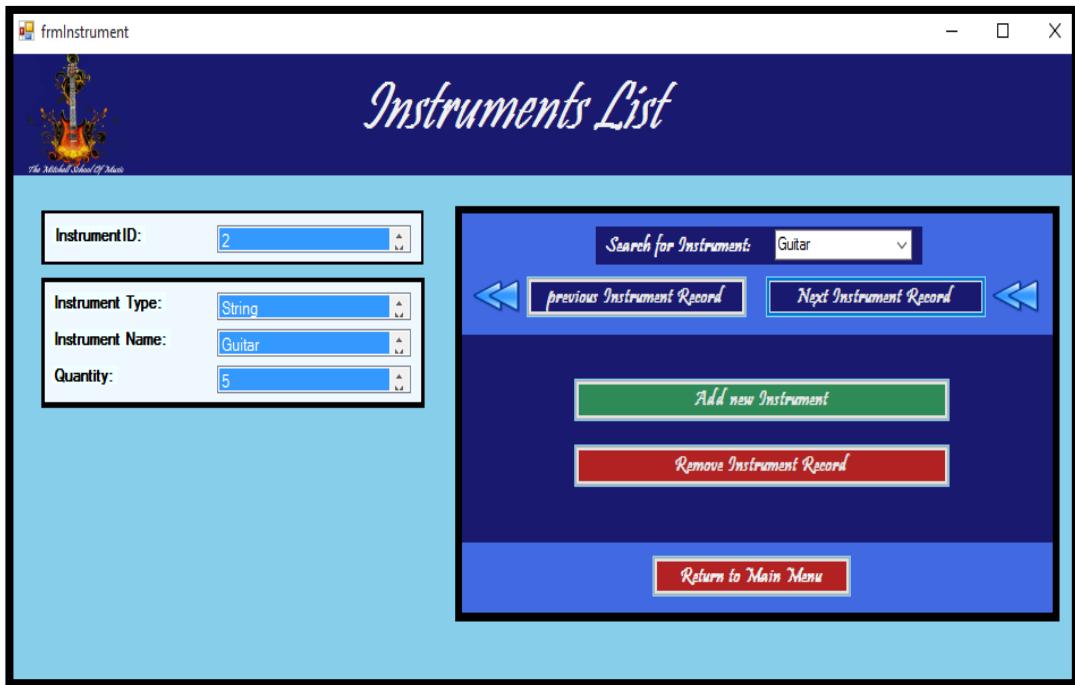
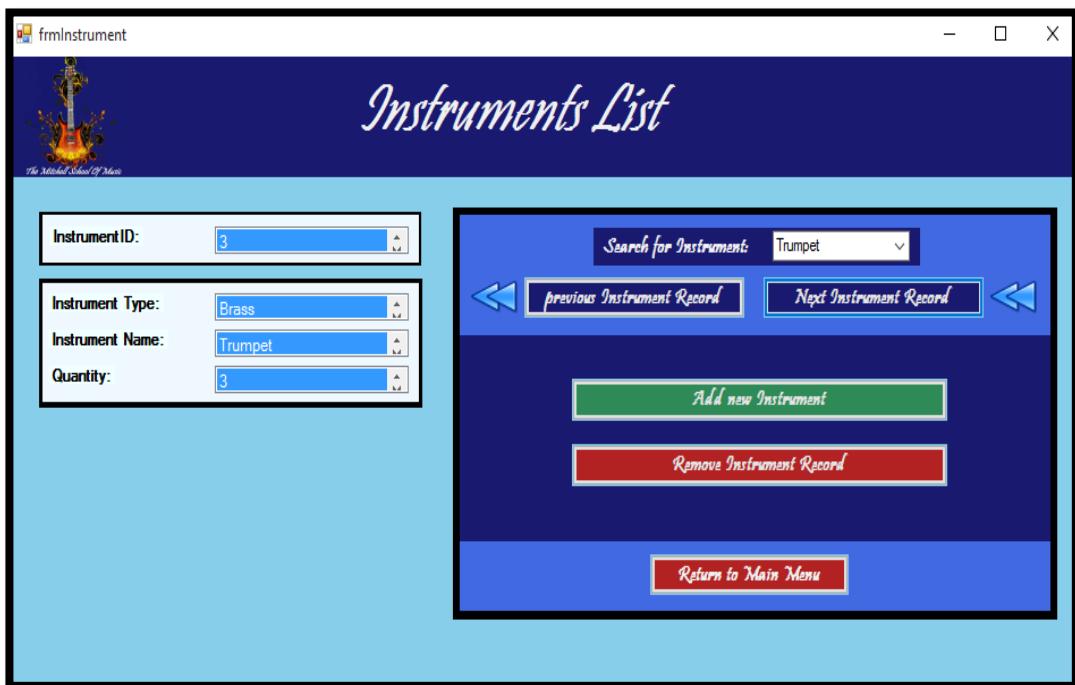
InstrumentID	Instrument Type	Instrument Name	Quantity
1	String	Violin	4
2	String	Guitar	5
3	Brass	Trumpet	3
4	Wood Wind	Flute	8
5	Percussion	Bass Drum	2
6	Keyboard	Piano	2
7	Wood Wind	Clarinet	4
8	Brass	Trombone	3
9	Percussion	Xylophone	5
10	Vocal	Singing	1
NULL	NULL	NULL	NULL

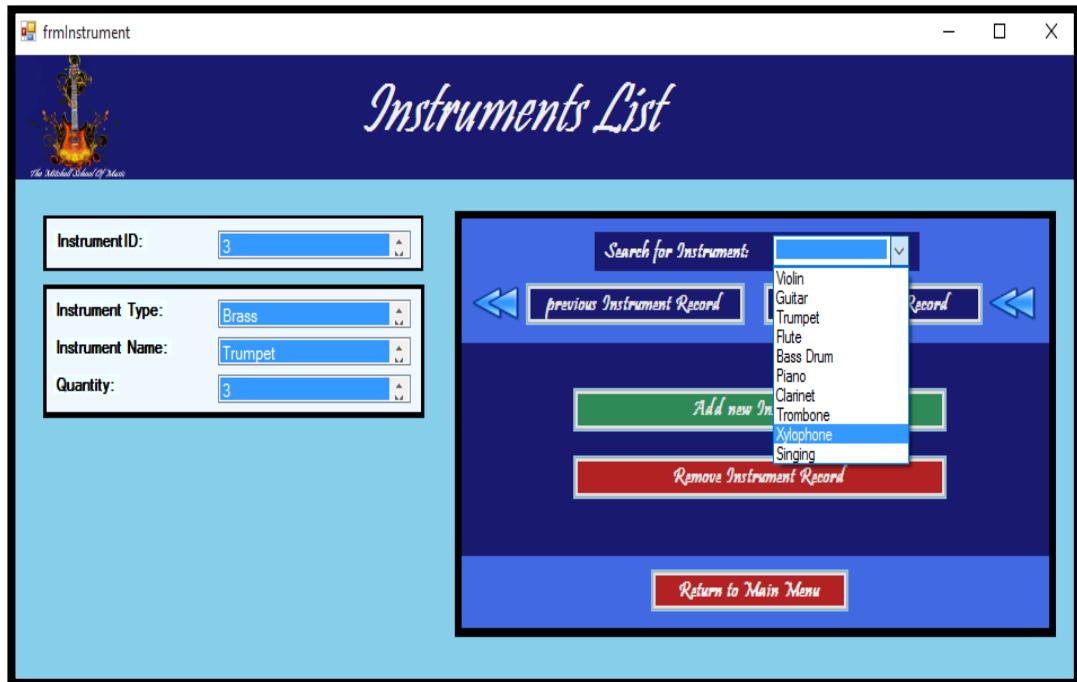
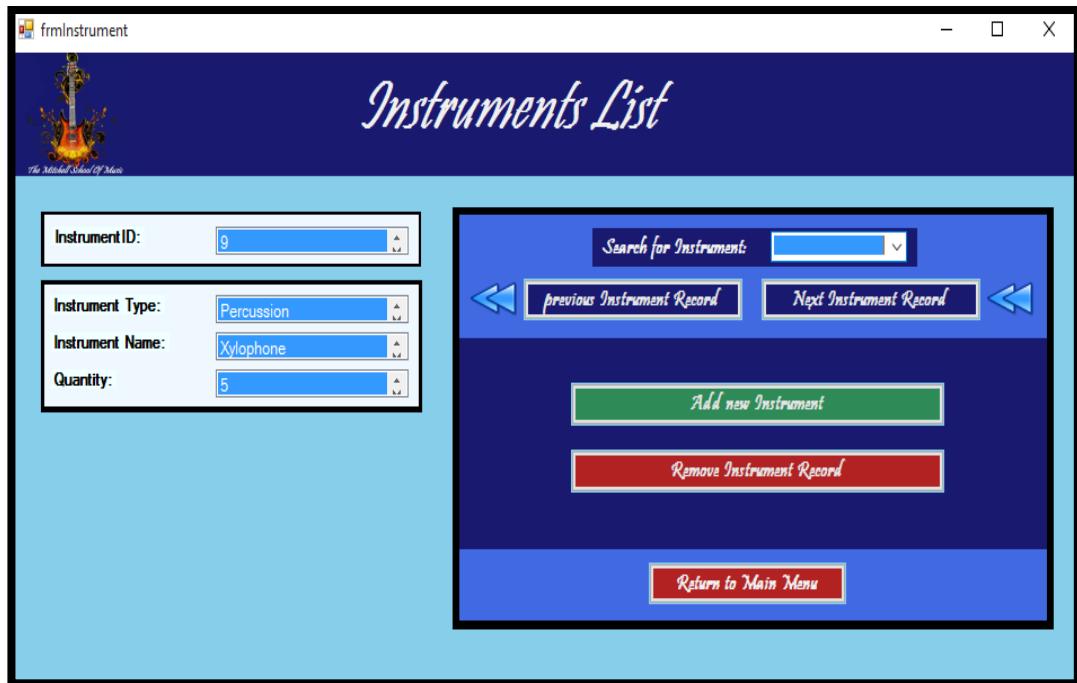
2.6[A][1] Screenshot B

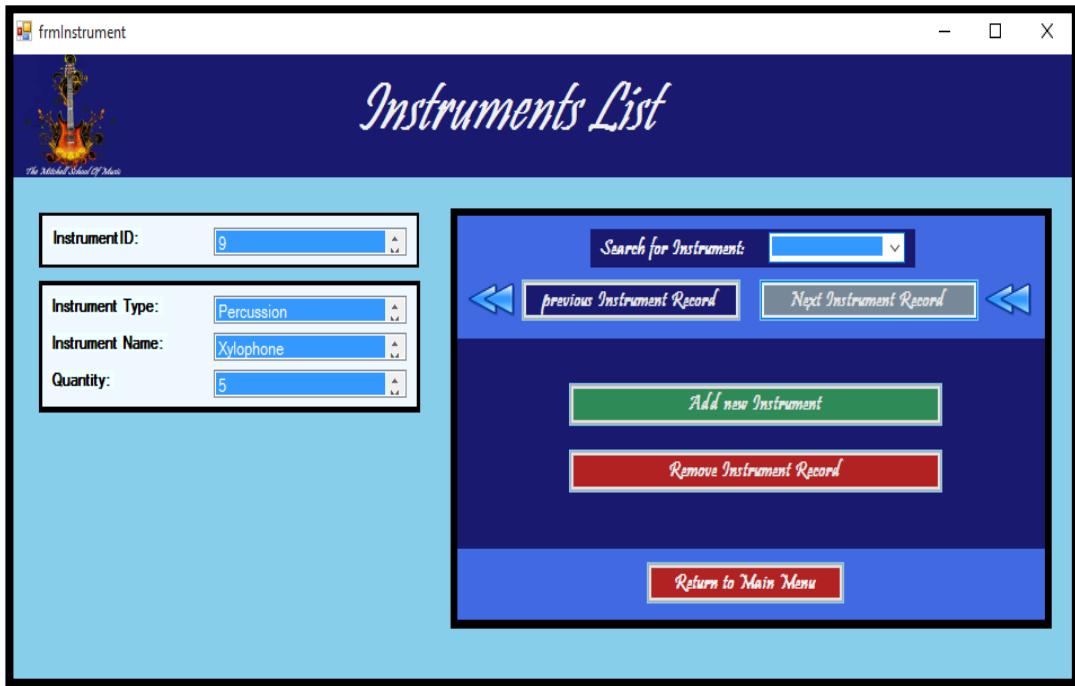
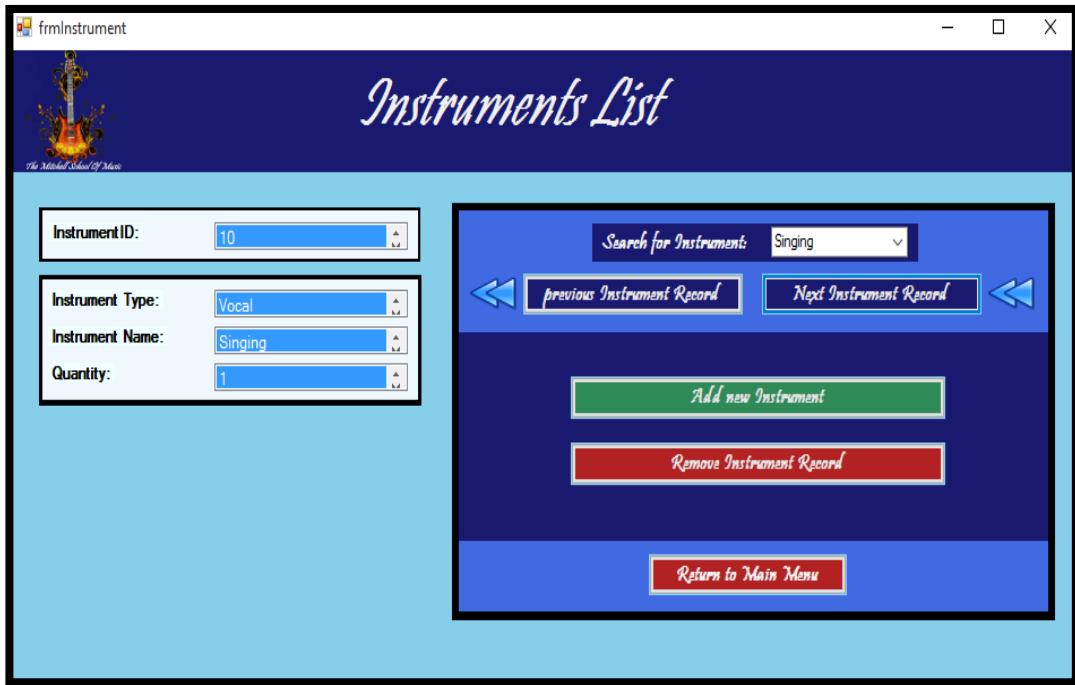


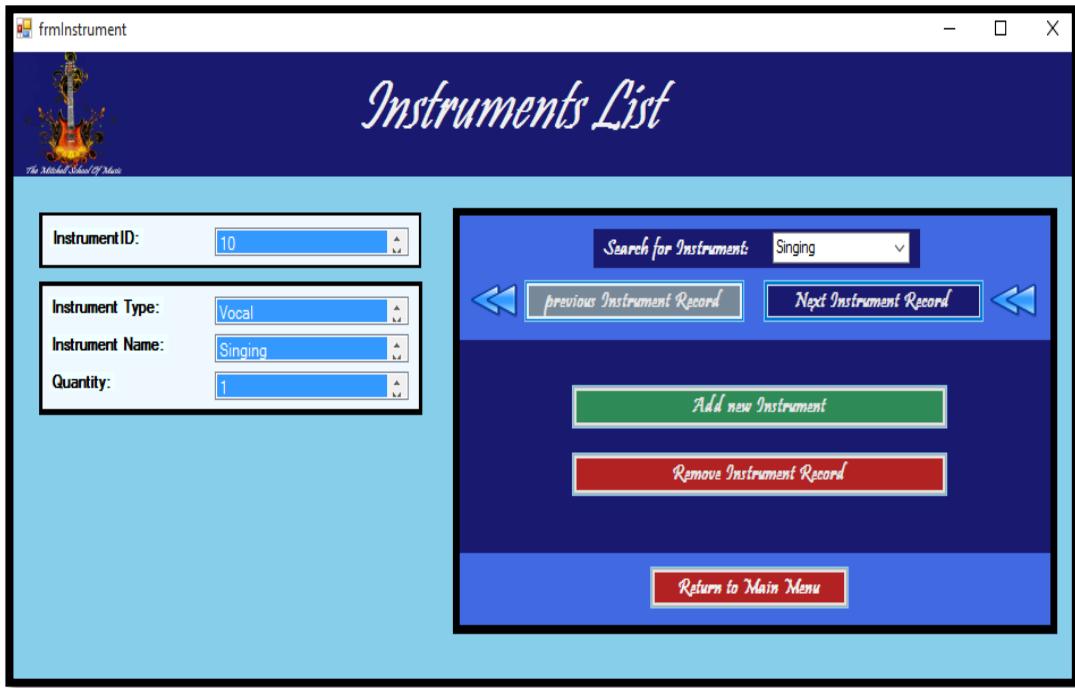
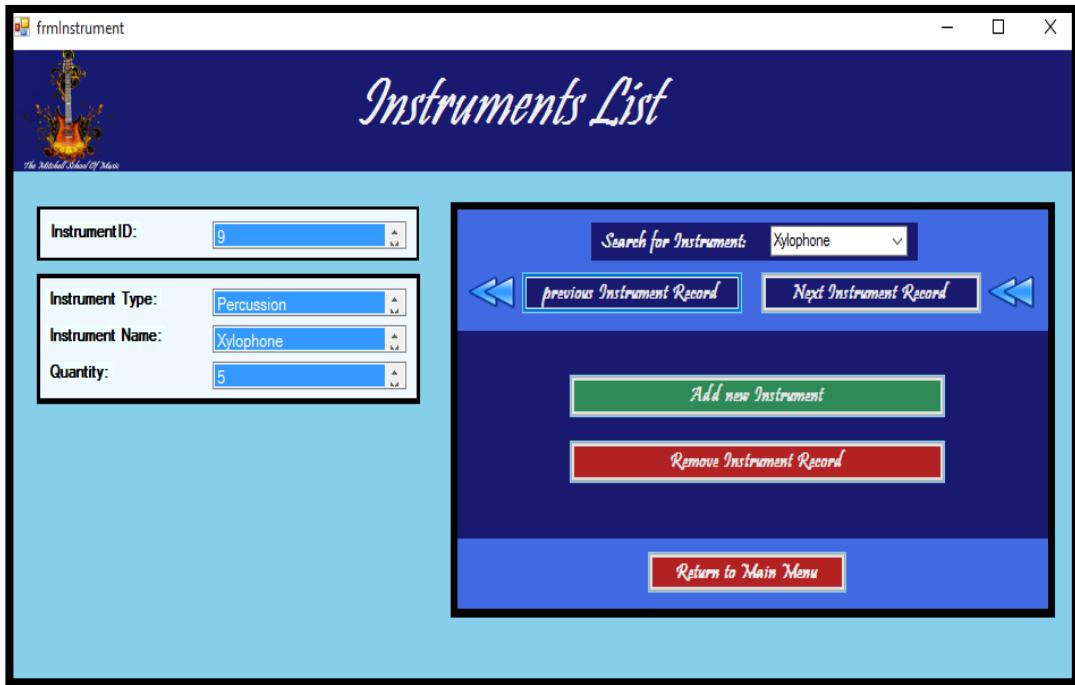
The application window title is "frmInstrument". The main title bar says "Instruments List". The interface includes:

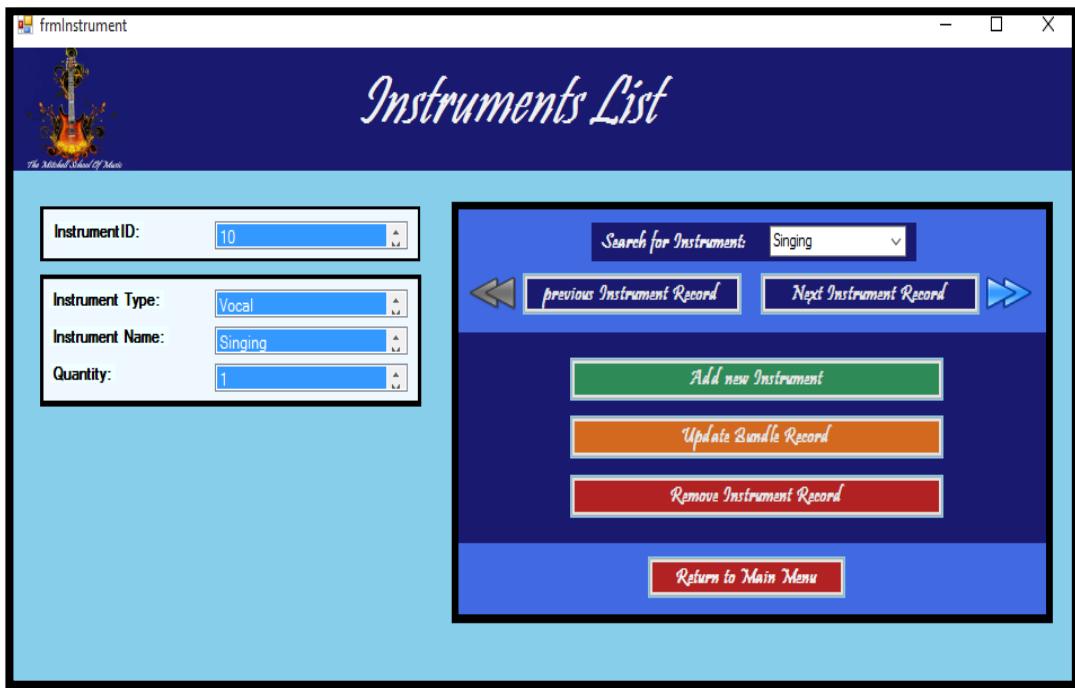
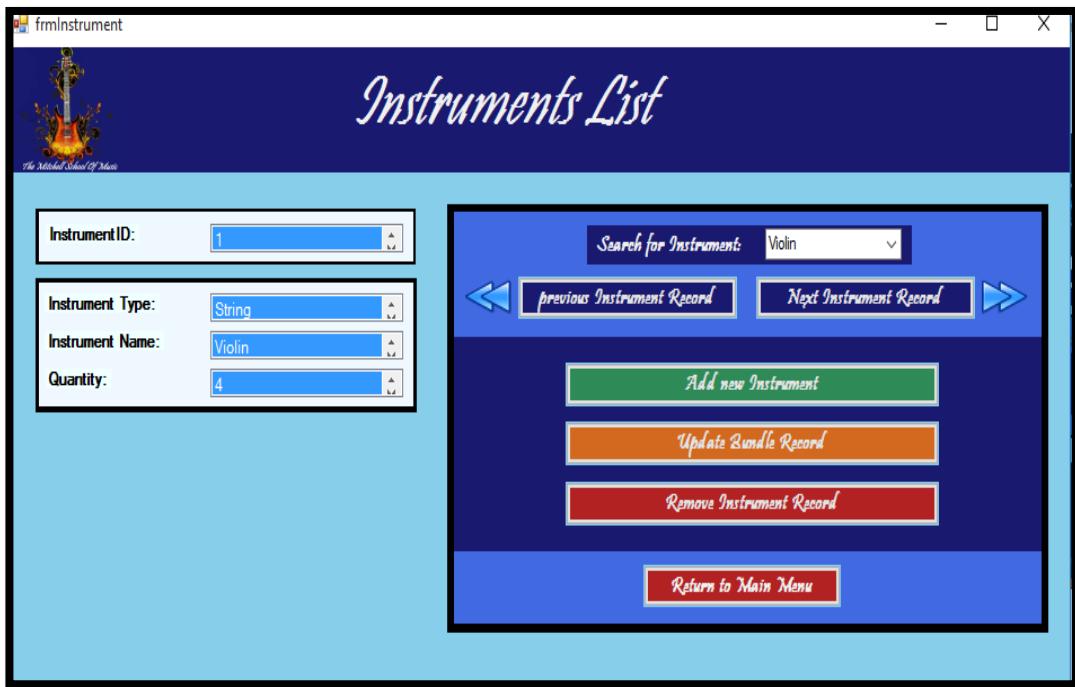
- A logo for "The Musical School Of Music" featuring a stylized guitar.
- An "Instrument ID" dropdown menu set to "1".
- An "Instrument Type" dropdown menu set to "String".
- An "Instrument Name" dropdown menu set to "Violin".
- An "Quantity" dropdown menu set to "4".
- A search bar labeled "Search for Instrument: Violin".
- Navigation buttons: "previous Instrument Record" and "Next Instrument Record".
- Action buttons: "Add new Instrument" (green), "Remove Instrument Record" (red), and "Return to Main Menu" (blue).

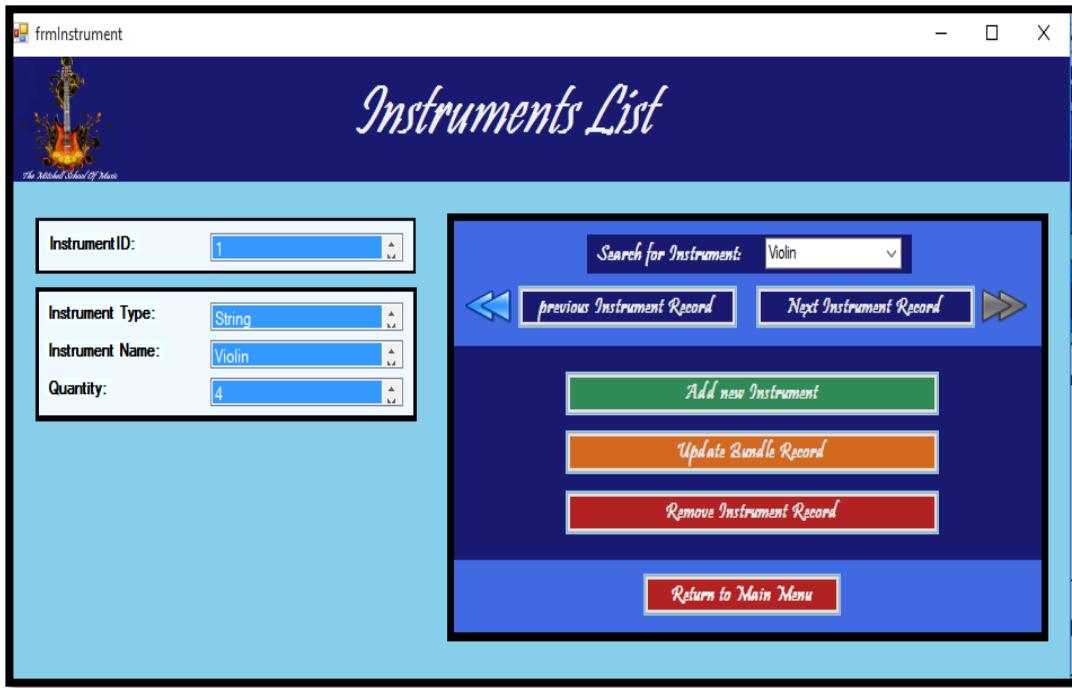
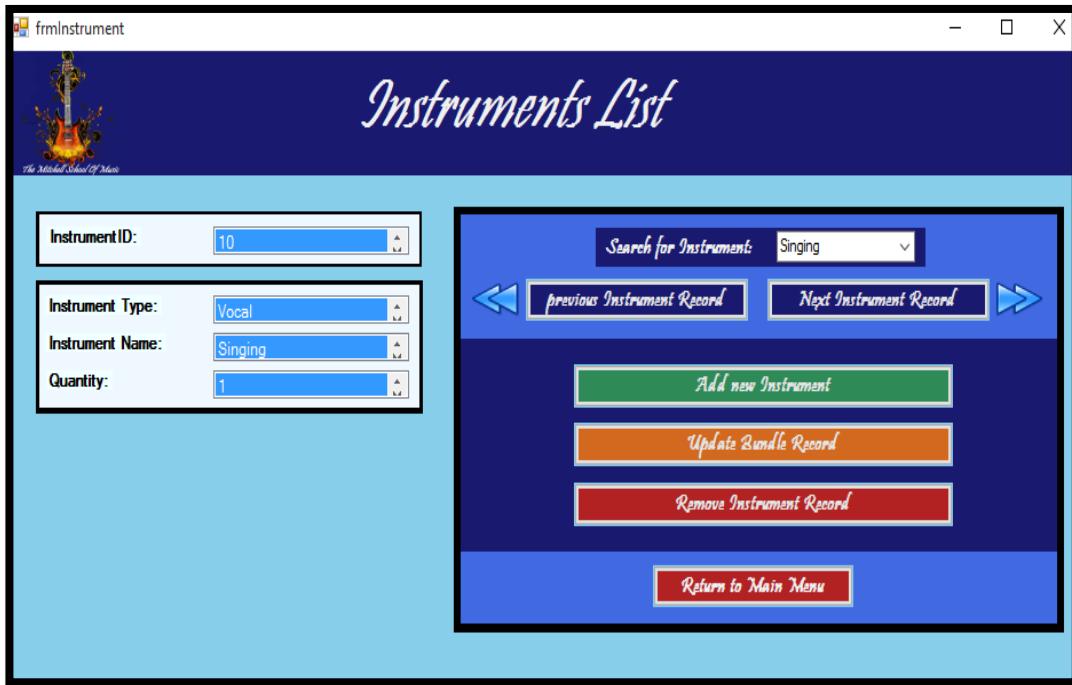
2.6[A][1] Screenshot C**2.6[A][1] Screenshot D**

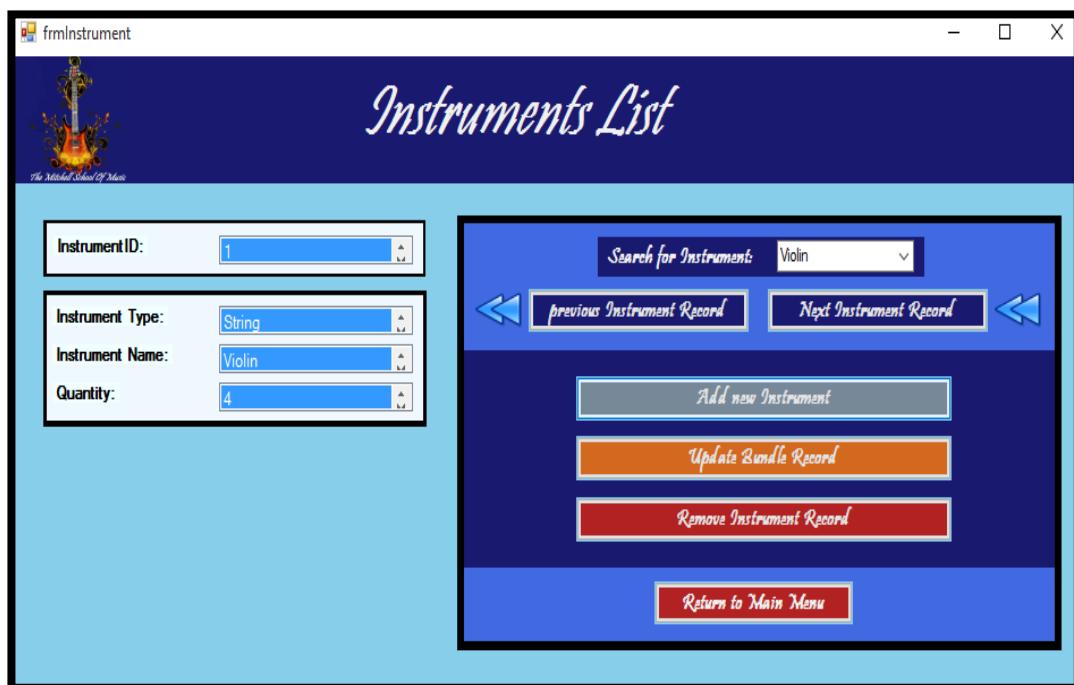
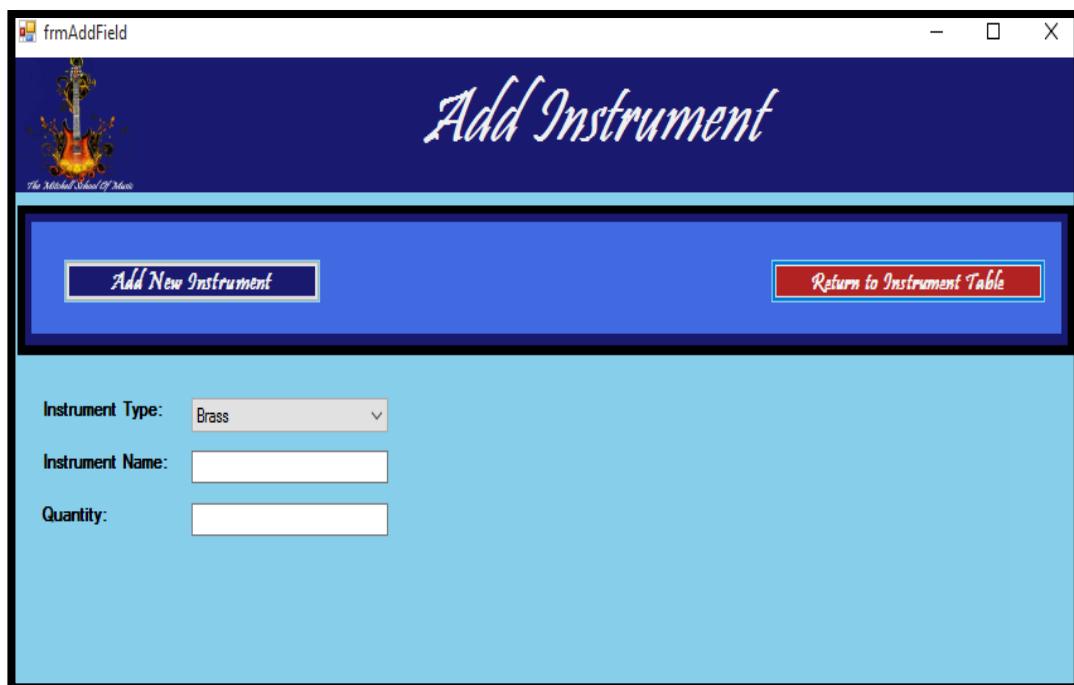
*Process [B]) Record Navigation***2.6[B][1] – Instrument Search****2.6[B][1] Screenshot A****2.6[B][1] Screenshot B**

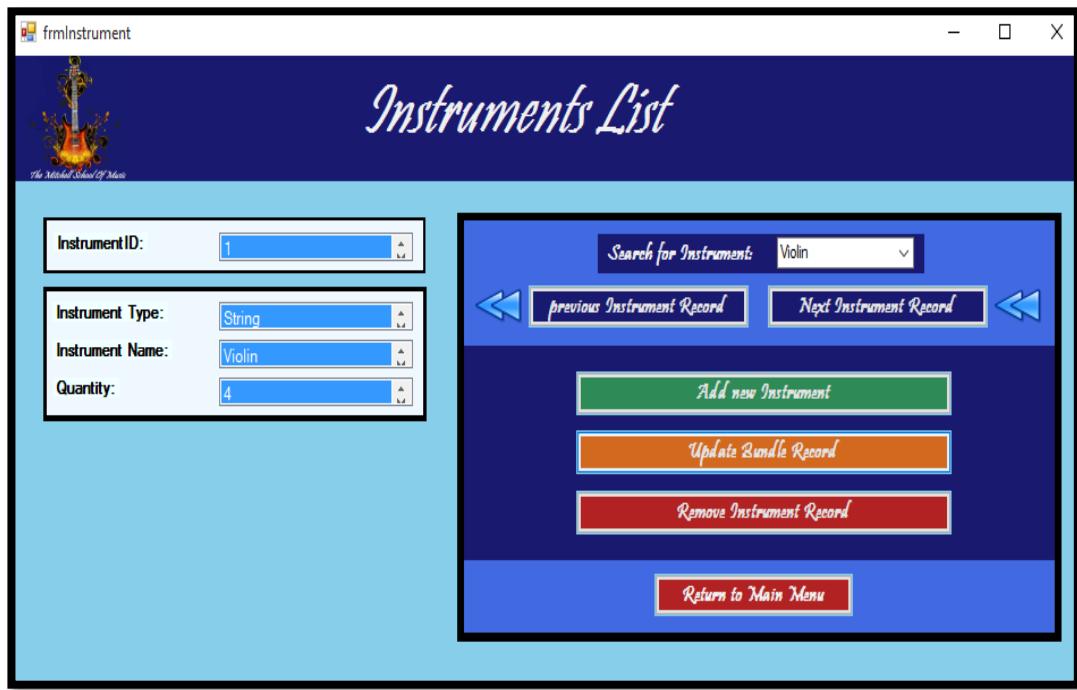
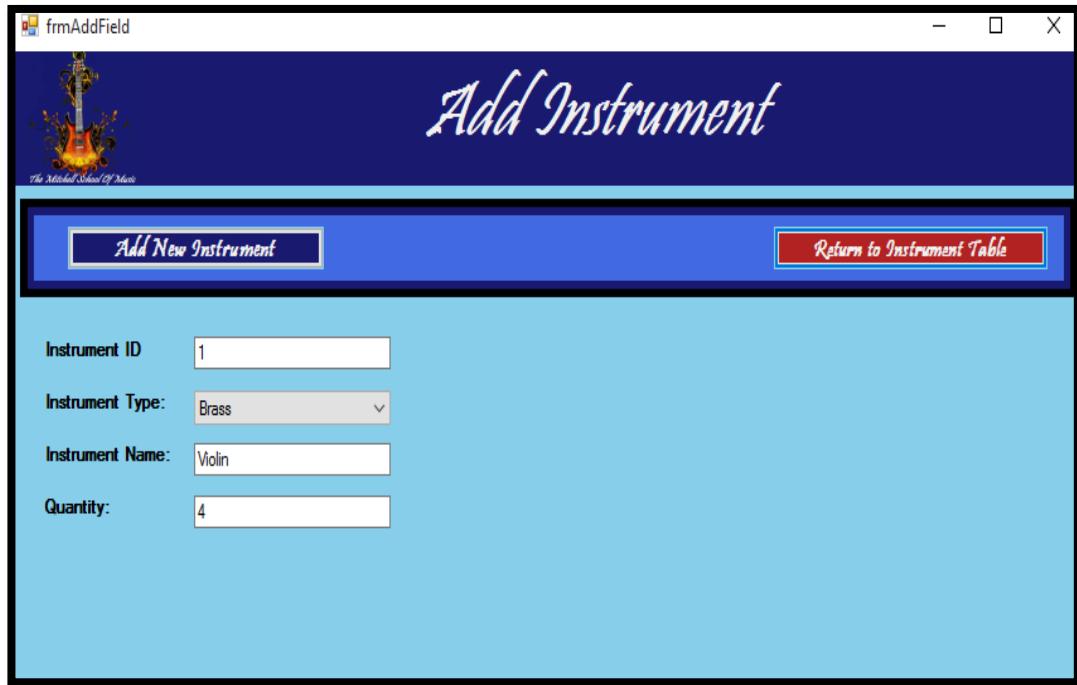
2.6[B][2] – Next Instrument**2.6[B][2] Screenshot A****2.6[B][2] Screenshot B**

2.6[B][3] – Previous Instrument**2.6[B][3] Screenshot A****2.6[B][3] Screenshot B**

2.6[B][4] – First Instrument**2.6[B][4] Screenshot A****2.6[B][4] Screenshot B**

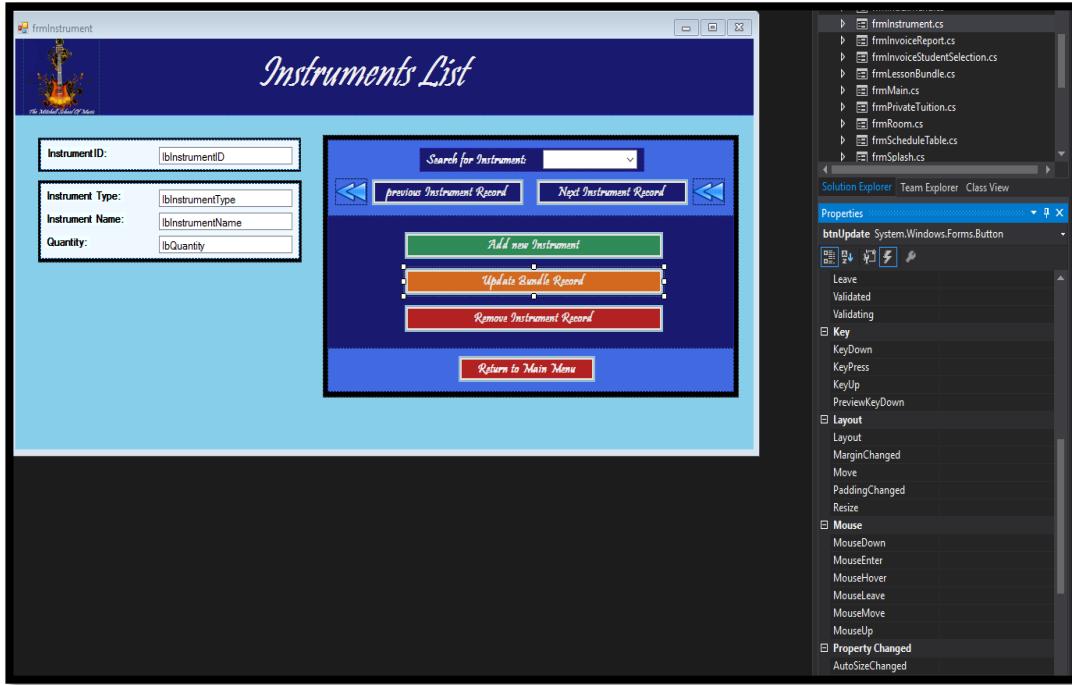
2.6[B][5] – Last Instrument**2.6[B][5] Screenshot A****2.6[B][5] Screenshot B**

*Process [C]) Form Navigation***2.6[C][1] – Add New Instrument Record****2.6[C][1] Screenshot A****2.6[C][1] Screenshot B**

2.6[C][2] – Update Instrument Record**2.6[C][2] Screenshot A****2.6[C][2] Screenshot B**

2.6[C][2][Corrective Action] – Update Instrument Record

2.6[C][Corrective Action] Screenshot A



2.6[C][Corrective Action] Screenshot B

```

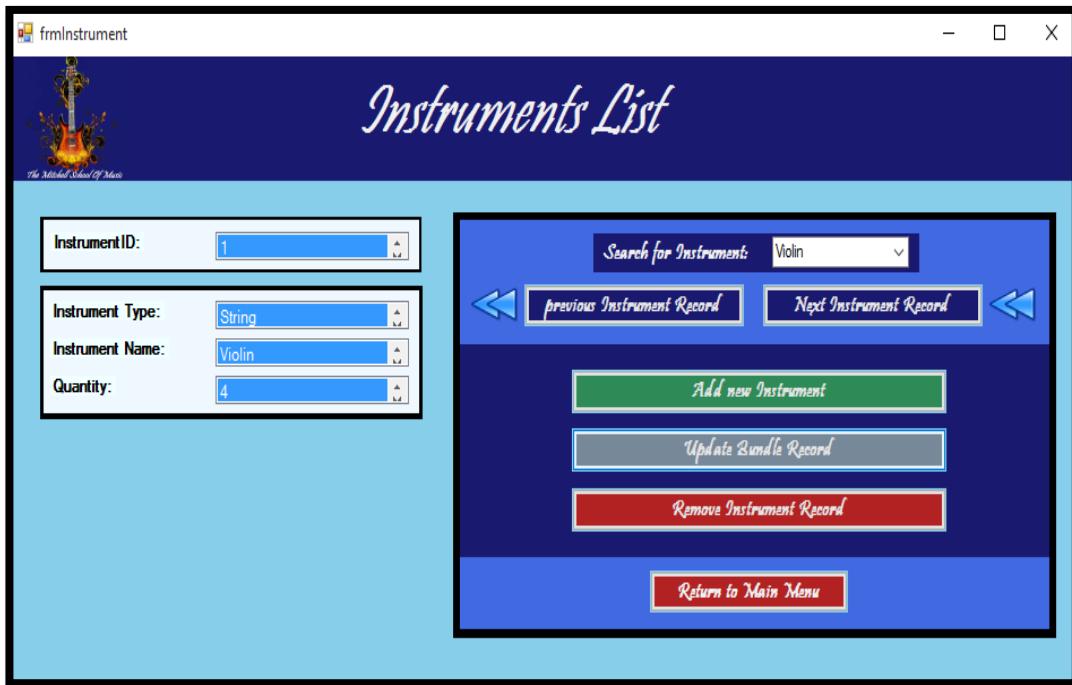
        }
    }

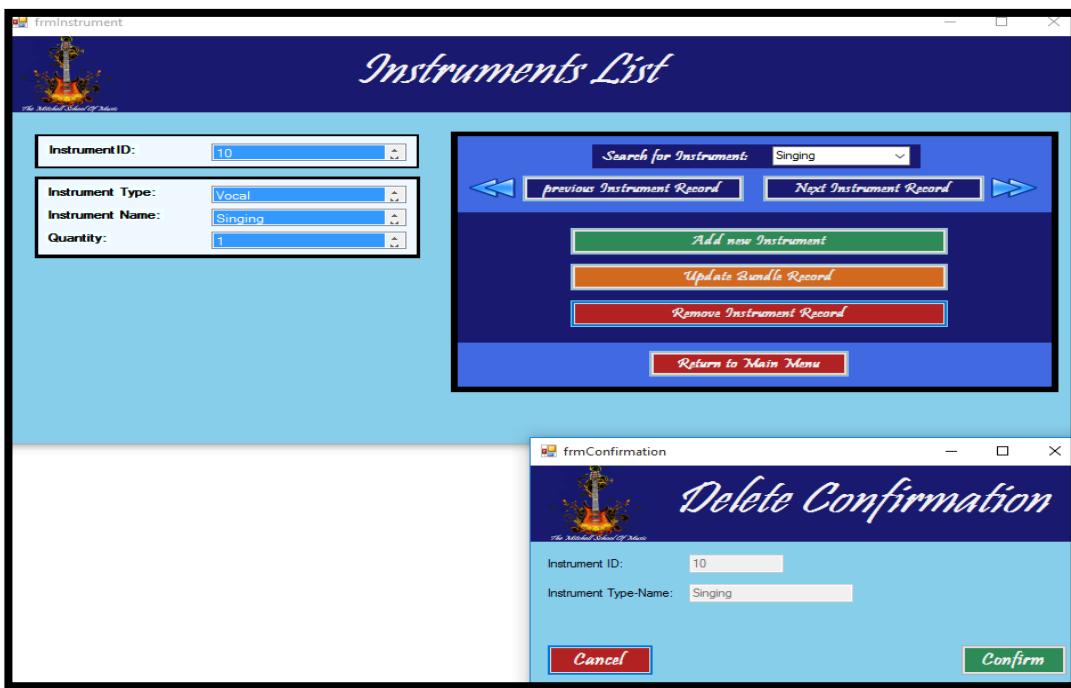
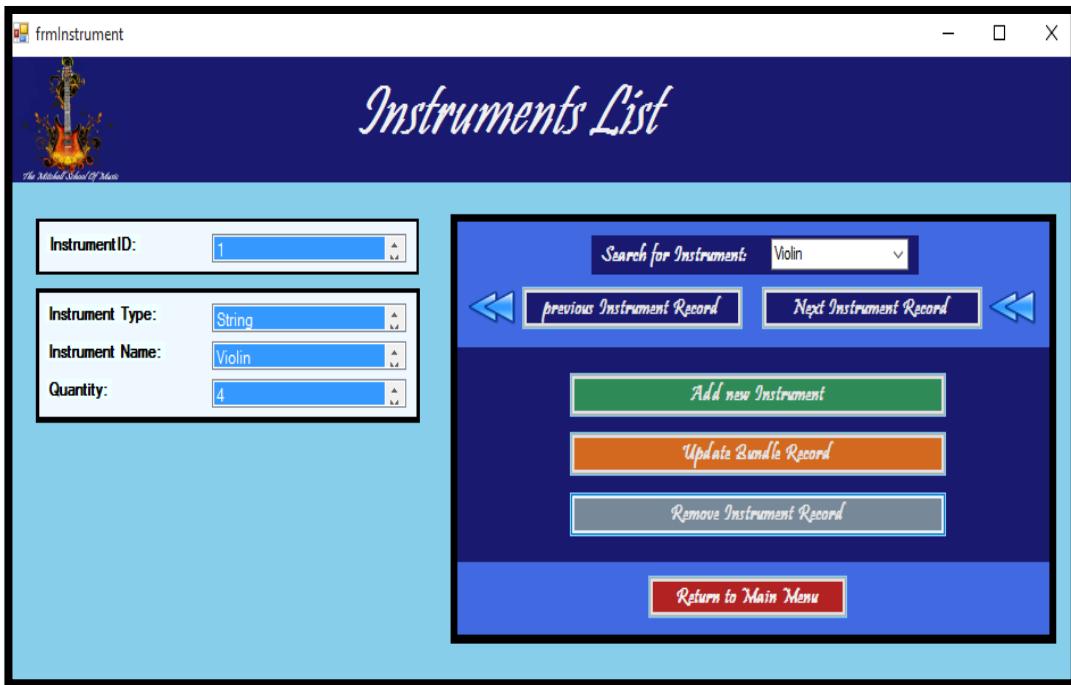
    // FIRST / LAST RECORD
    reference
    private void pbFirstRecord_Click(object sender, EventArgs e)
    {
        lbInstrumentID.SelectedValue = Instrument_IDs[0];
    }
    reference
    private void pbLastRecord_Click(object sender, EventArgs e)
    {
        lbInstrumentID.SelectedValue = Instrument_IDs[Instrument_IDs.Count() - 1];
    }

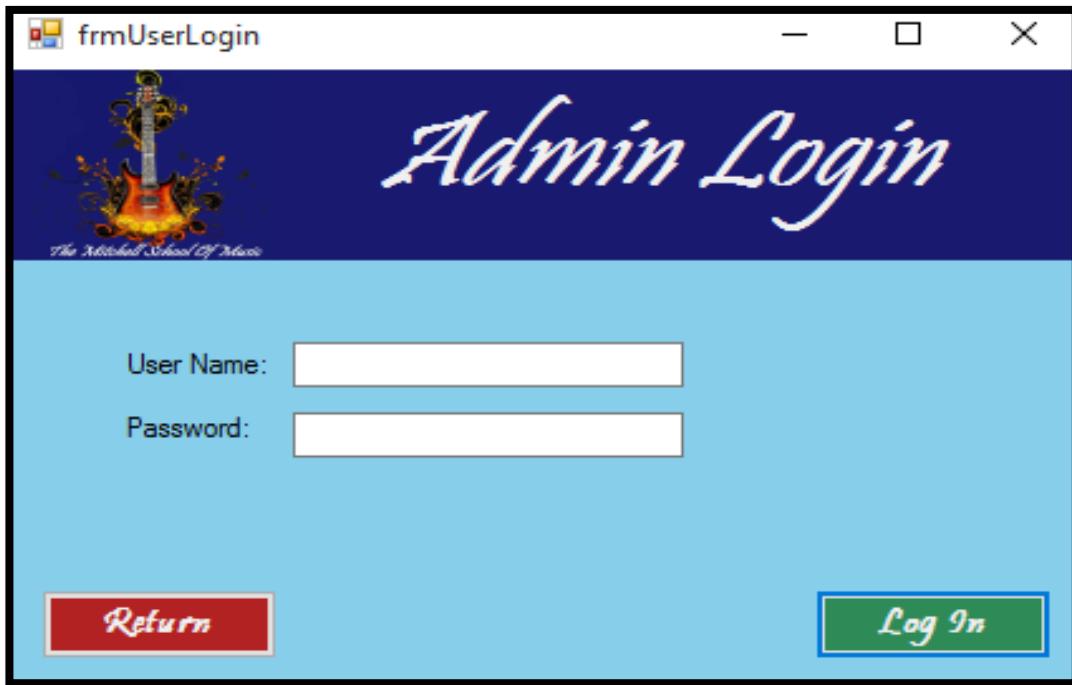
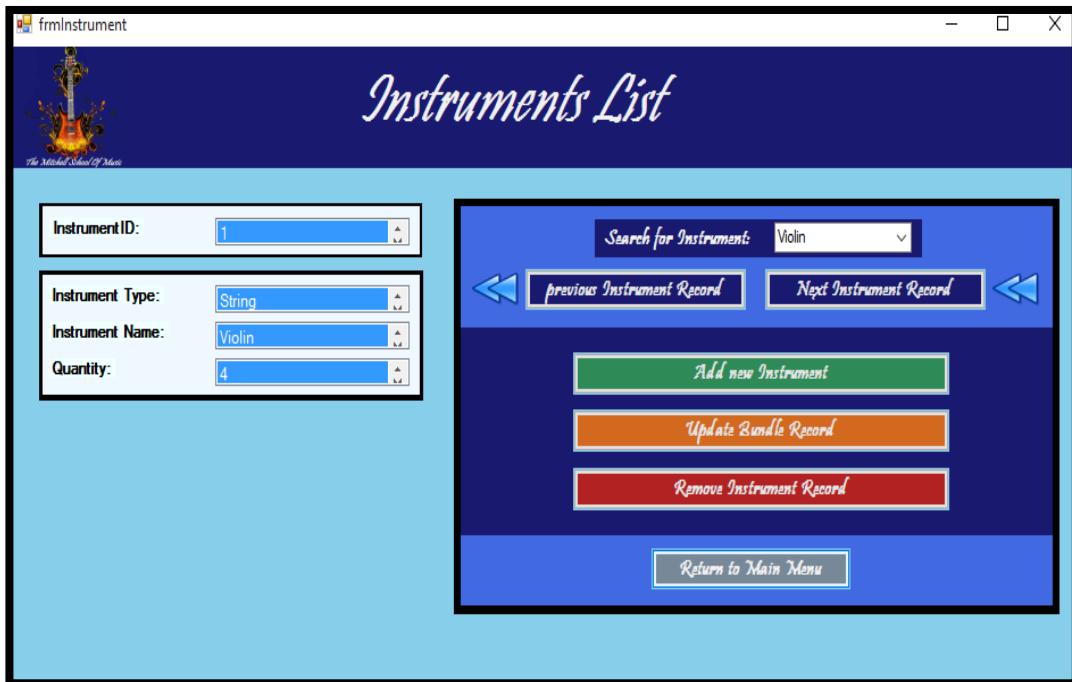
    // UPDATE BUTTON
    reference
    private void btnUpdate_MouseEnter(object sender, EventArgs e)
    {
        btnUpdate.BackColor = Color.LightSlateGray;
        this.Cursor = Cursors.Hand;
    }
    reference
    private void btnUpdate_MouseLeave(object sender, EventArgs e)
    {
        btnUpdate.BackColor = Color.Chocolate;
        this.Cursor = Cursors.Arrow;
    }
}
]

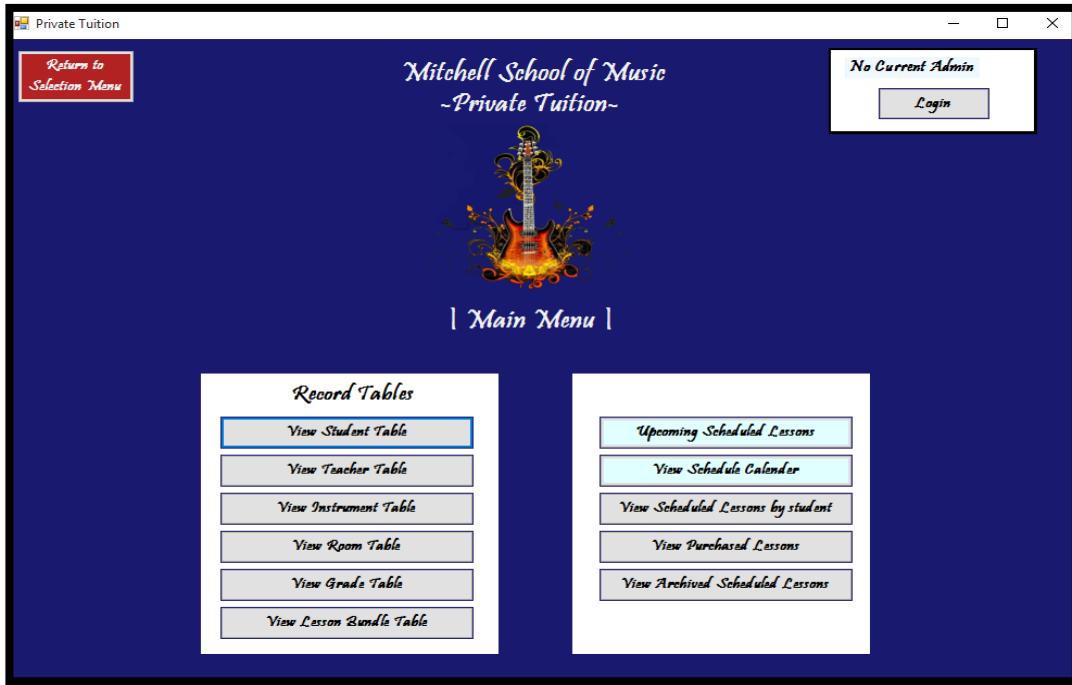
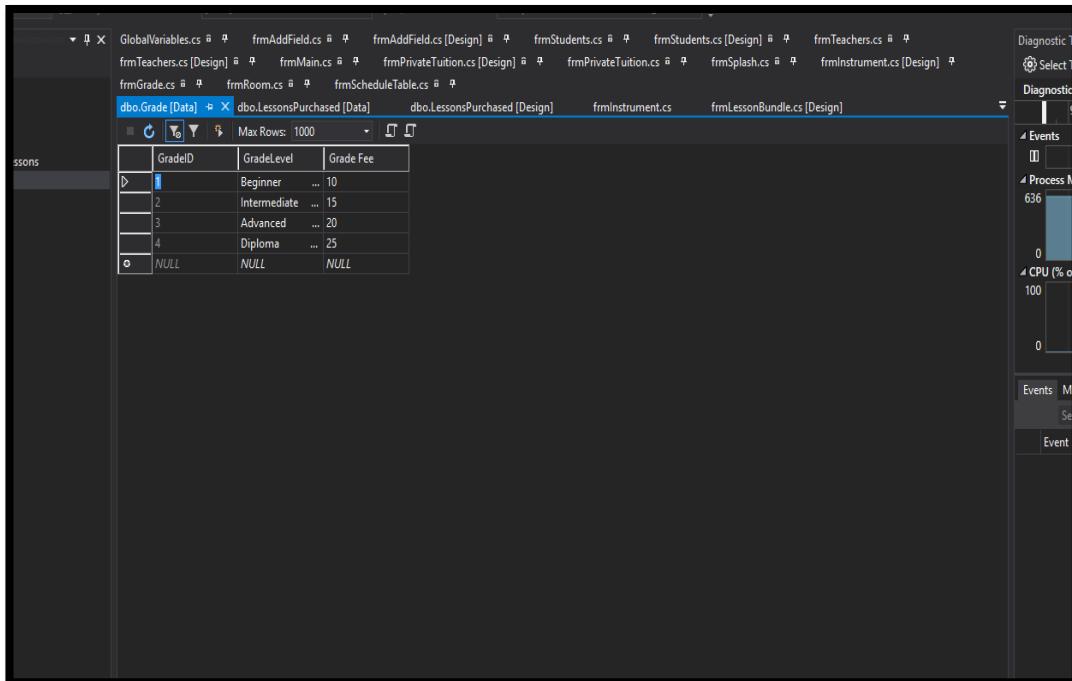
```

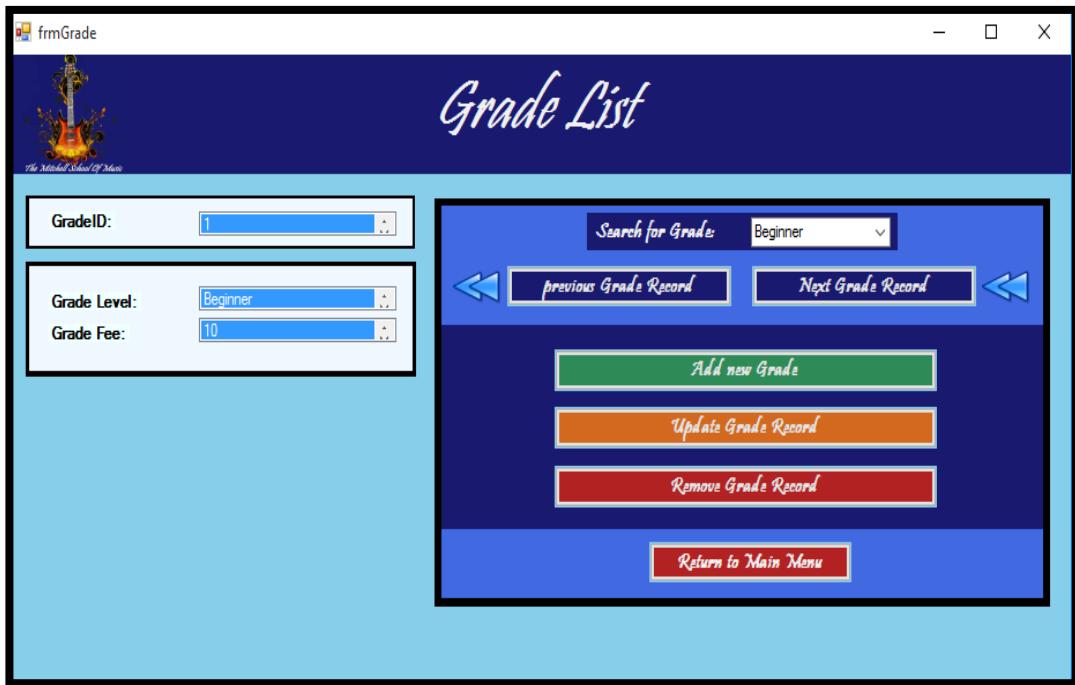
The screenshot shows the Visual Studio code editor displaying C# code. The code defines several event handlers for buttons: "pbFirstRecord_Click", "pbLastRecord_Click", and "btnUpdate". The "btnUpdate" handler contains logic to change the button's background color and cursor based on the mouse position. The code is part of a class definition, indicated by the opening brace at the bottom.

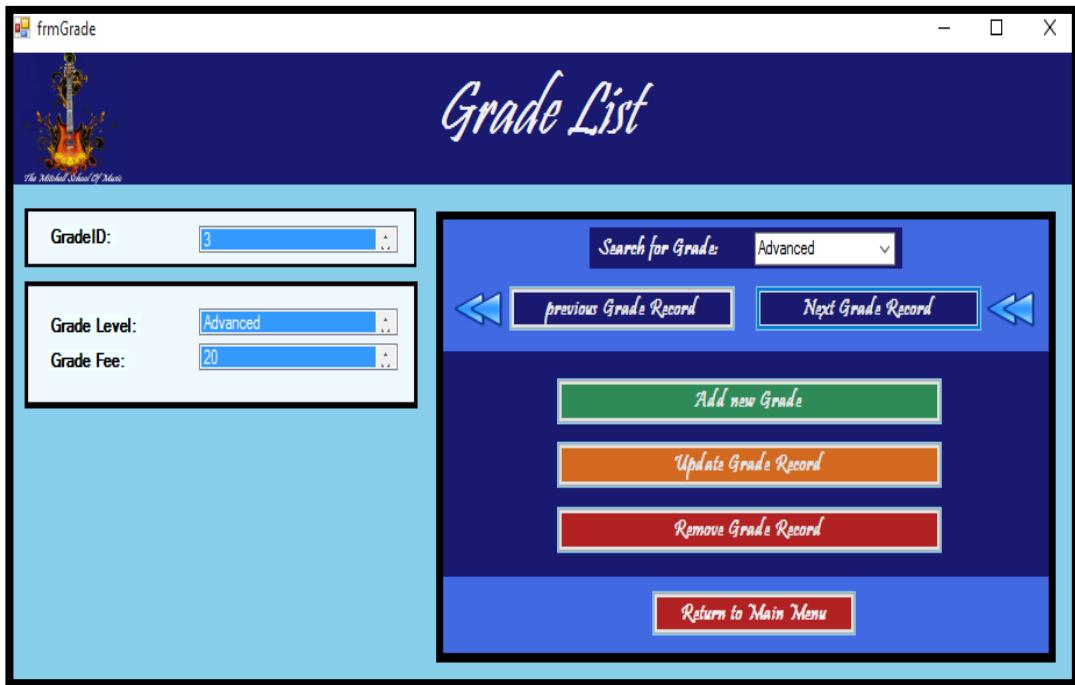
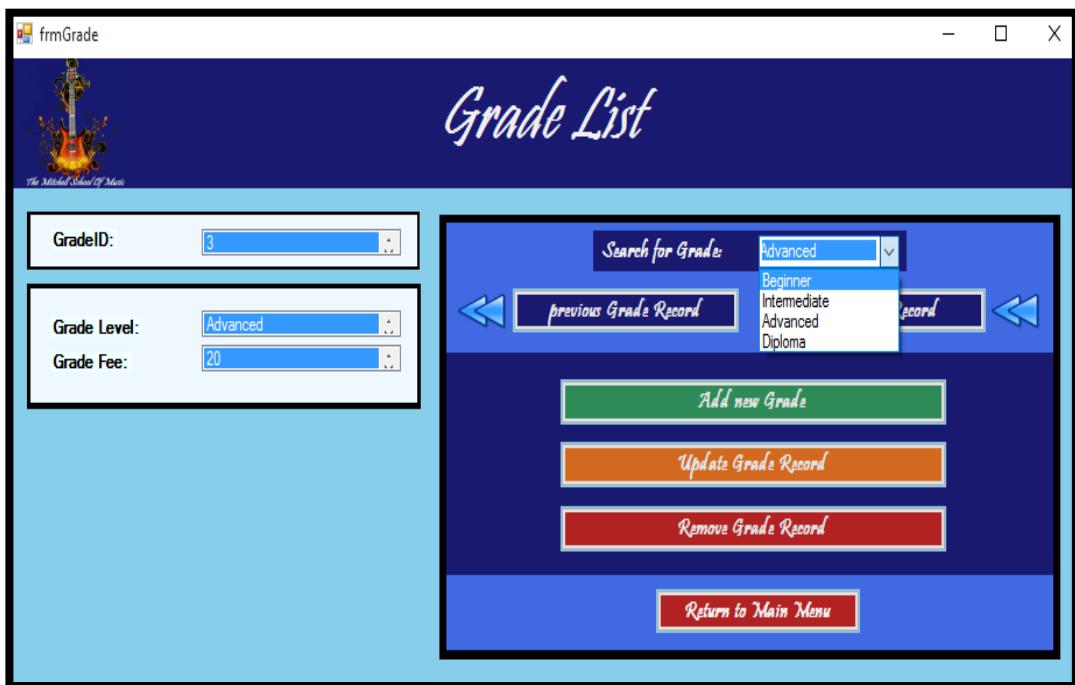
2.6[C][Corrective Action] Screenshot C**2.6[C][3] – Delete Instrument Record [User Logged In]****2.6[C][3] Screenshot A**

2.6[C][3] Screenshot B**2.6[C][4] – Delete Instrument Record [No User Logged In]****2.6[C][4] Screenshot A**

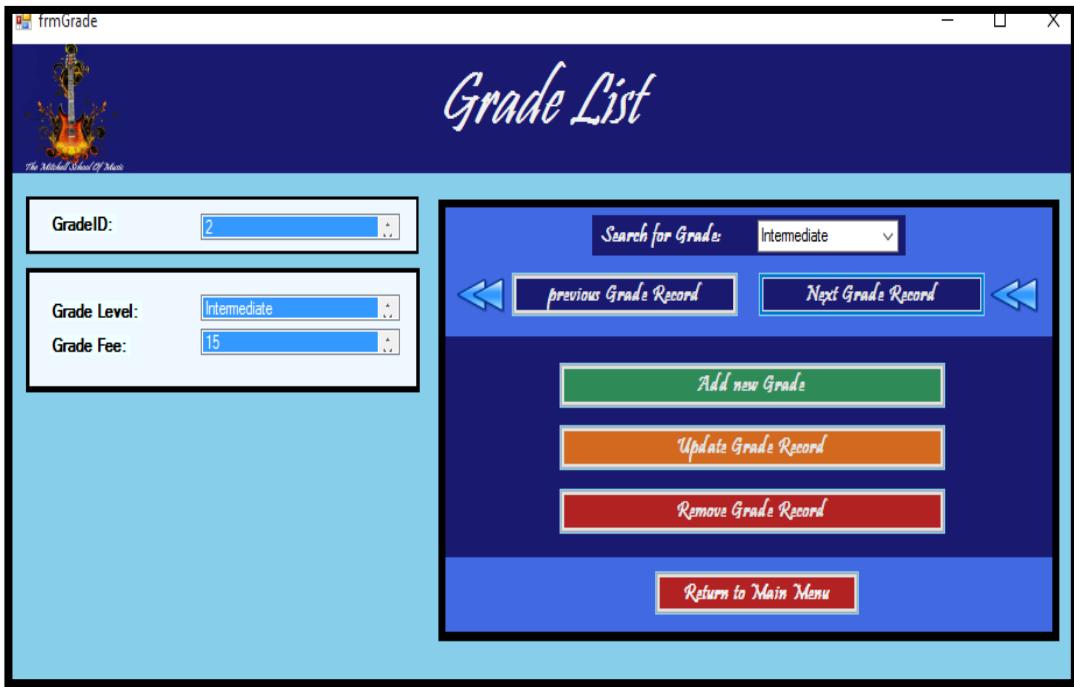
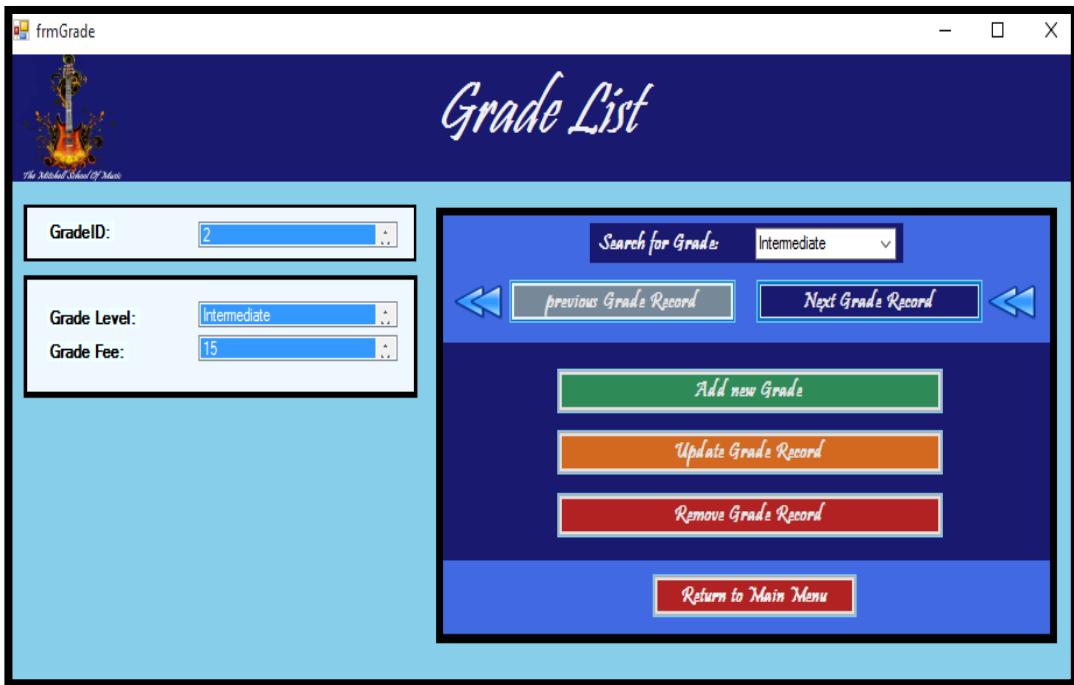
2.6[C][4] Screenshot B**2.6[C][5] – Return To Main Menu****2.6[C][5] Screenshot A**

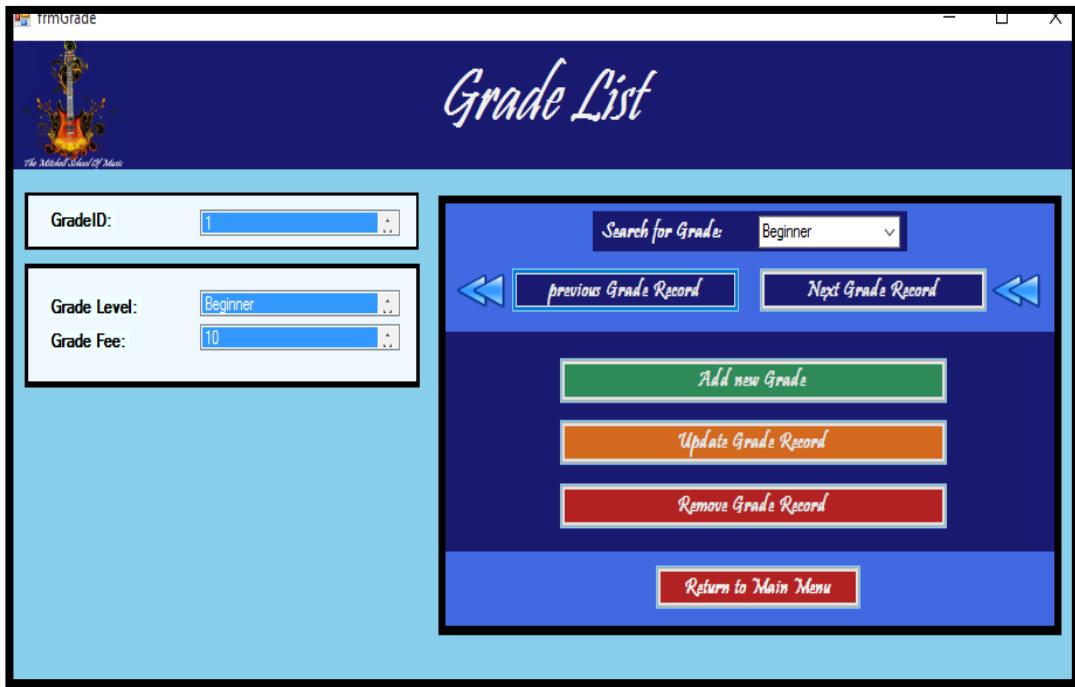
2.6[C][5] Screenshot B**2.7) Grade Form***Process [A]) Information Display***2.7[A][1] – Grade Information****2.7[A][1] Screenshot A**

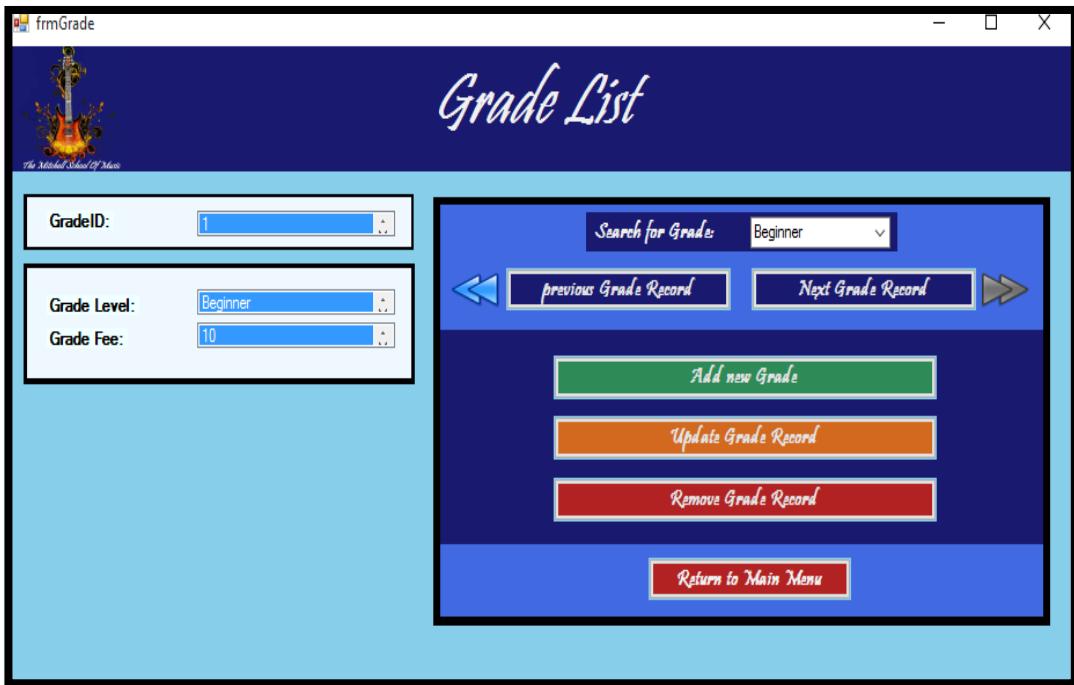
2.7[A][1] Screenshot B**2.7[A][1] Screenshot C**

2.7[A][1] Screenshot D**Process [B]) Record Navigation****2.7[B][1] – Grade Search****2.7[B][1] Screenshot A**

2.7[B][1] Screenshot B**2.7[B][2] – Next Grade****2.7[B][2] Screenshot A**

2.7[B][2] Screenshot B**2.7[B][3] – Previous Grade****2.7[B][3] Screenshot A**

2.7[B][3] Screenshot B**2.7[B][4] – First Grade****2.7[B][4] Screenshot A**

2.7[B][4] Screenshot B**2.7[B][5] – Last Grade****2.7[B][5] Screenshot A**

2.7[B][5] Screenshot B

frmGrade

Grade List

The Musical School Of Music

GradeID: 4

Grade Level: Diploma

Grade Fee: 25

Search for Grade: Diploma

previous Grade Record Next Grade Record

Add new Grade

Update Grade Record

Remove Grade Record

Return to Main Menu

Process [C]) Form Navigation**2.7[C][1] – Add New Grade Record****2.7[C][1] Screenshot A**

frmGrade

Grade List

The Musical School Of Music

GradeID: 1

Grade Level: Beginner

Grade Fee: 10

Search for Grade: Beginner

previous Grade Record Next Grade Record

Add new Grade

Update Grade Record

Remove Grade Record

Return to Main Menu

2.7[C][1] Screenshot B**2.7[C][2] – Update Grade Record****2.7[C][2] Screenshot A**

2.7[C][2] Screenshot B**2.7[C][2][Corrective Action] – Update Grade Record****2.7[C][Corrective Action] Screenshot A**

Screenshot A shows the Visual Studio IDE with the code editor open to the frmAddField.cs file. The code handles updates to different tables based on the previous form:

```

if (GlobalVariables.PreviousForm == "InstrumentTable")
{
    iMain.Text = "Update Teacher Record";
    iBFieldDetailA.Text = "Teacher ID";
    iBFieldDetailB.Text = GlobalVariables.Teacher.FirstName;
    iBFieldDetailC.Text = GlobalVariables.Teacher.Surname;
    iBFieldDetailD.Text = GlobalVariables.Teacher.Address;
    iBFieldDetailE.Text = GlobalVariables.Teacher.Town;
    iBFieldDetailF.Text = GlobalVariables.Teacher.Postcode;
    iBFieldDetailG.Text = GlobalVariables.Teacher.Email;
    iBFieldDetailH.Text = GlobalVariables.Teacher.ContactNumber;
    iBFieldDetailI.Text = GlobalVariables.Teacher.Specialisation;
    iBFieldDetailJ.Text = GlobalVariables.Teacher.RoomID;
}

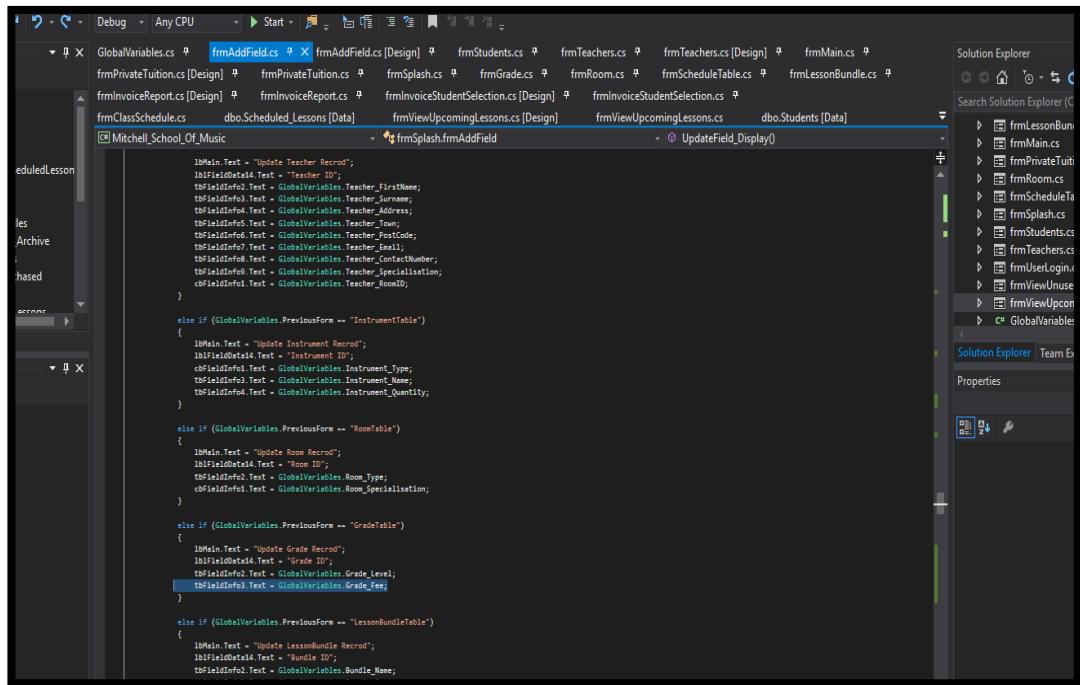
else if (GlobalVariables.PreviousForm == "InstrumentTable")
{
    iMain.Text = "Update Instrument Record";
    iBFieldDetailA.Text = "Instrument ID";
    iBFieldDetailB.Text = GlobalVariables.Instrument.Type;
    iBFieldDetailC.Text = GlobalVariables.Instrument_Name;
    iBFieldDetailD.Text = GlobalVariables.Instrument_Quality;
}

else if (GlobalVariables.PreviousForm == "RoomTable")
{
    iMain.Text = "Update Room Record";
    iBFieldDetailA.Text = "Room ID";
    iBFieldDetailB.Text = GlobalVariables.Room_Type;
    iBFieldDetailC.Text = GlobalVariables.Room_Specialisation;
}

else if (GlobalVariables.PreviousForm == "GradeTable")
{
    iMain.Text = "Update Grade Record";
    iBFieldDetailA.Text = "Grade ID";
    iBFieldDetailB.Text = GlobalVariables.Grade_Level;
    iBFieldDetailC.Text = GlobalVariables.Grade_Fee;
}

else
{
    iMain.Text = "Update LessonBundle Record";
    iBFieldDetailA.Text = "Bundle ID";
    iBFieldDetailB.Text = GlobalVariables.Bundle_Bear;
    iBFieldDetailC.Text = GlobalVariables.Bundle_Cost;
    iBFieldDetailD.Text = GlobalVariables.Bundle_Discount;
}

```

2.7[C][Corrective Action] Screenshot B


```

        lbMain.Text = "Update Teacher Record";
        lbFieldInfo14.Text = "Teacher ID";
        tbFieldInfo02.Text = GlobalVariables.Teacher_FirstName;
        tbFieldInfo03.Text = GlobalVariables.Teacher_Surname;
        tbFieldInfo04.Text = GlobalVariables.Teacher_Address;
        tbFieldInfo05.Text = GlobalVariables.Teacher_PostCode;
        tbFieldInfo06.Text = GlobalVariables.Teacher_Email;
        tbFieldInfo07.Text = GlobalVariables.Teacher_ContactNumber;
        tbFieldInfo08.Text = GlobalVariables.Teacher_Specialisation;
        cbFieldInfo01.Text = GlobalVariables.Teacher_RoomID;

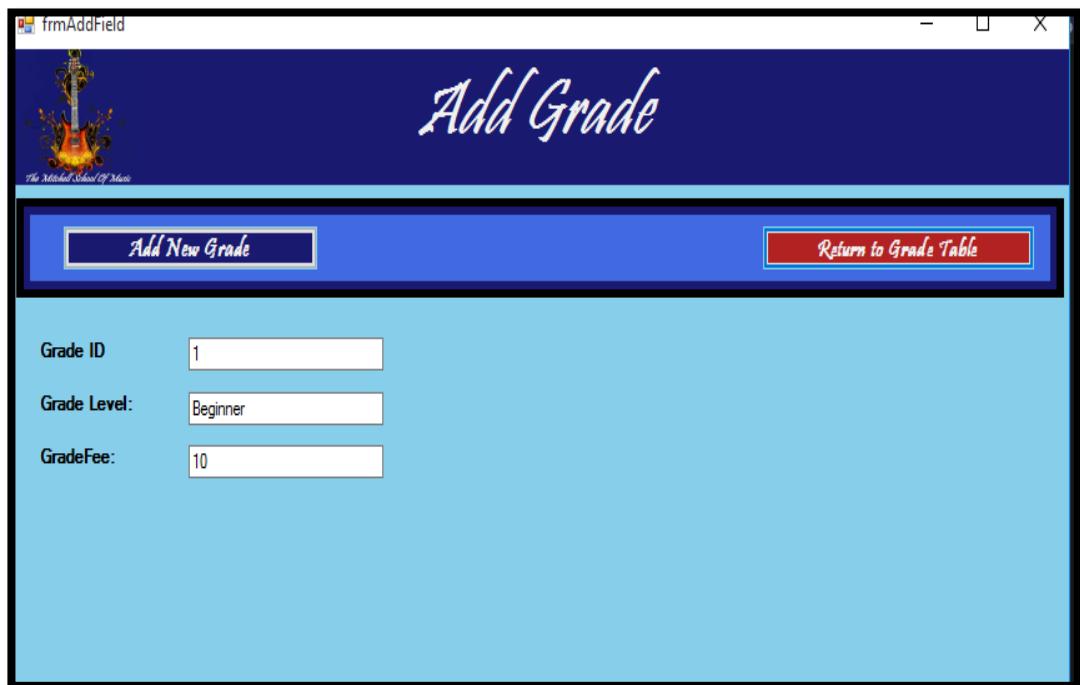
    } else if (GlobalVariables.PreviousForm == "InstrumentTable")
    {
        lbMain.Text = "Update Instrument Record";
        lbFieldInfo14.Text = "Instrument ID";
        cbFieldInfo01.Text = GlobalVariables.Instrument_Type;
        tbFieldInfo02.Text = GlobalVariables.Instrument_Name;
        tbFieldInfo03.Text = GlobalVariables.Instrument_Quantity;
    }

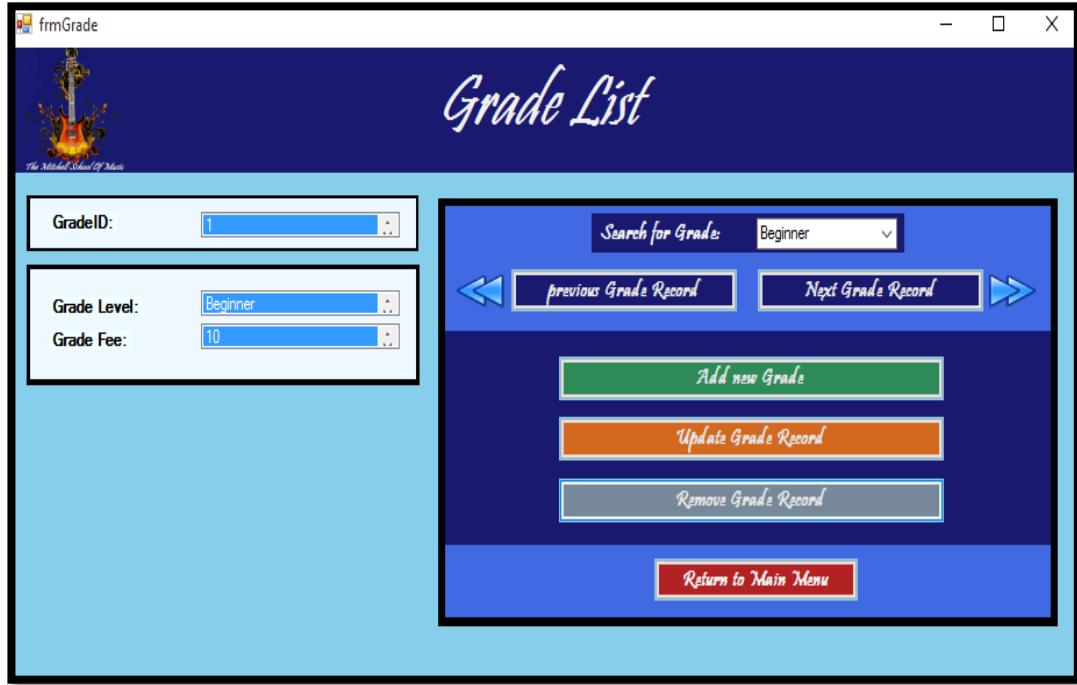
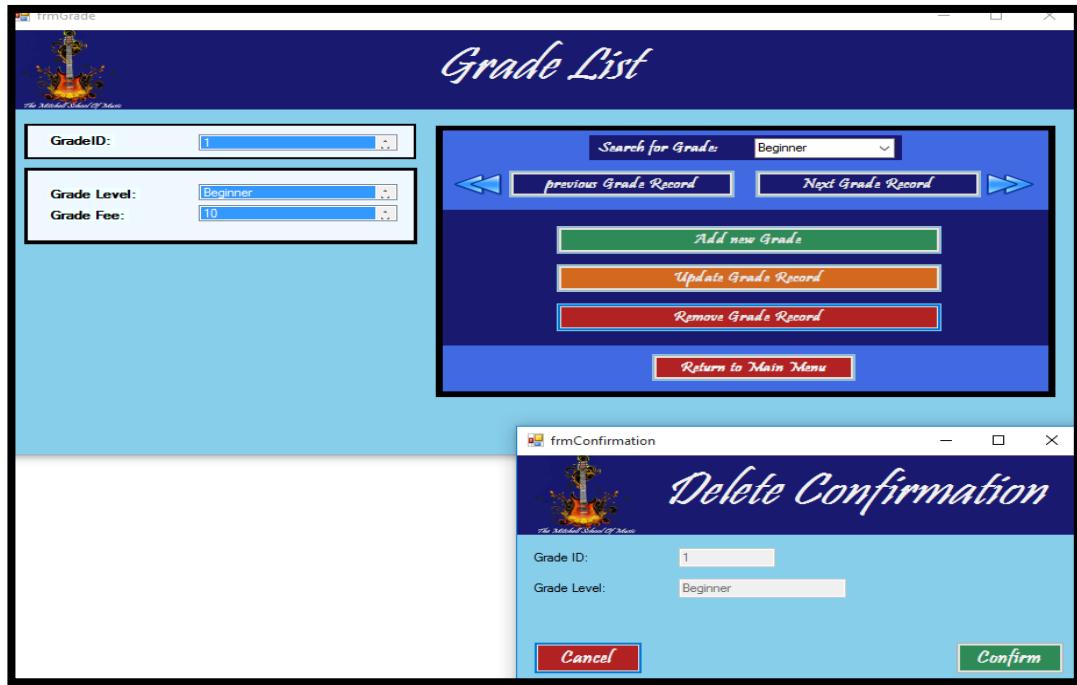
    else if (GlobalVariables.PreviousForm == "RoomTable")
    {
        lbMain.Text = "Update Room Record";
        lbFieldInfo14.Text = "Room ID";
        tbFieldInfo02.Text = GlobalVariables.Room_Type;
        cbFieldInfo01.Text = GlobalVariables.Room_Specialisation;
    }

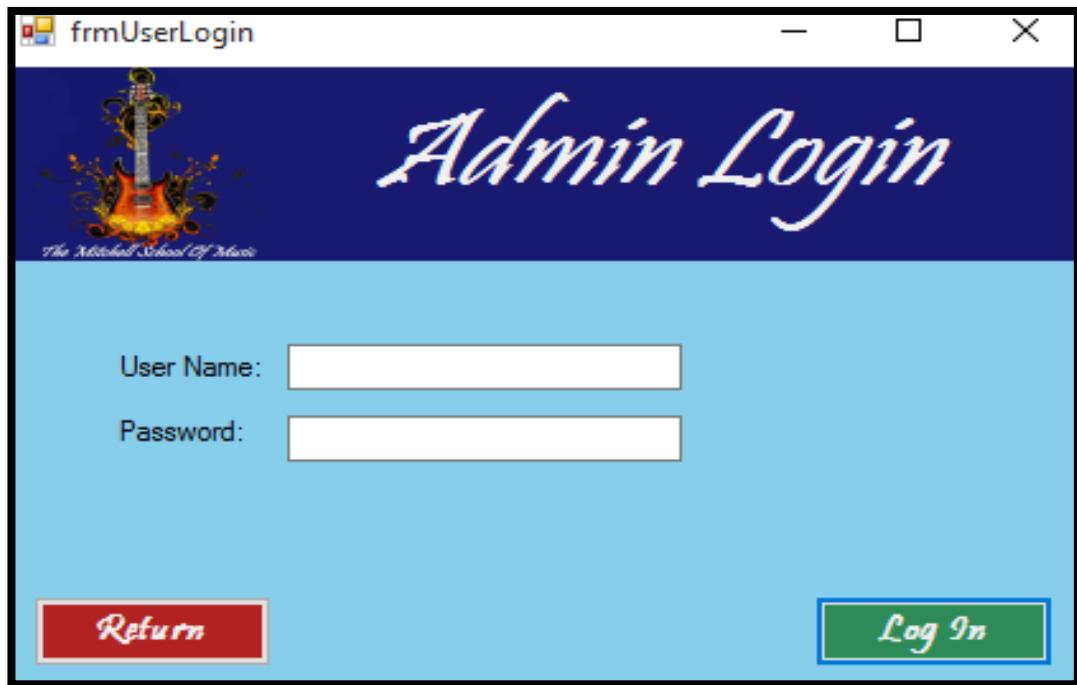
    else if (GlobalVariables.PreviousForm == "GradeTable")
    {
        lbMain.Text = "Update Grade Record";
        lbFieldInfo14.Text = "Grade ID";
        tbFieldInfo02.Text = GlobalVariables.Grade_Level;
        tbFieldInfo03.Text = GlobalVariables.Grade_Fee;
    }

    else if (GlobalVariables.PreviousForm == "LessonBundleTable")
    {
        lbMain.Text = "Update LessonBundle Record";
        lbFieldInfo14.Text = "Bundle ID";
        tbFieldInfo02.Text = GlobalVariables.Bundle_Name;
    }
}

```

2.7[C][Corrective Action] Screenshot C

2.7[C][3] – Delete Grade Record [User Logged In]**2.7[C][3] Screenshot A****2.7[C][3] Screenshot B**

2.7[C][4] – Delete Grade Record [No User Logged In]**2.7[C][4] Screenshot A****2.7[C][4] Screenshot B**

2.7[C][5] – Return To Main Menu**2.7[C][5] Screenshot A****2.7[C][5] Screenshot B**

2.8) Lesson Bundle Form***Process [A] Information Display******2.8[A][1] – Bundle Information******2.8[A][1] Screenshot A***

	LessonBundleID	Lesson Bundle	Bundle Cost	Multiplier (Dis...)
1	1	10 Lessons	... 10	1
2	2	20 Lessons	... 20	0.95
3	3	30 Lessons	... 30	0.9
	NULL	NULL	NULL	NULL

2.8[A][1] Screenshot B

frmLessonBundle

The Mitchell School Of Music

Lesson Bundle List

Lesson BundleID: 1

Lesson Bundle: 10 Lessons

Bundle Cost: 10

Multiplier(Discount): 1

Search for Bundle: 10 Lessons

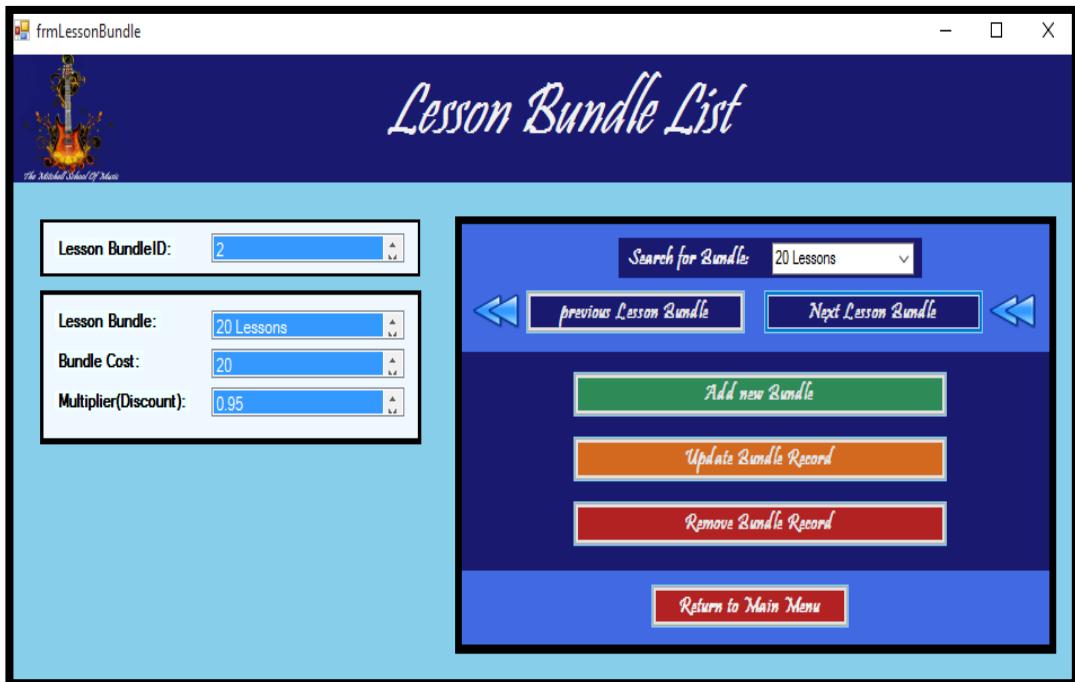
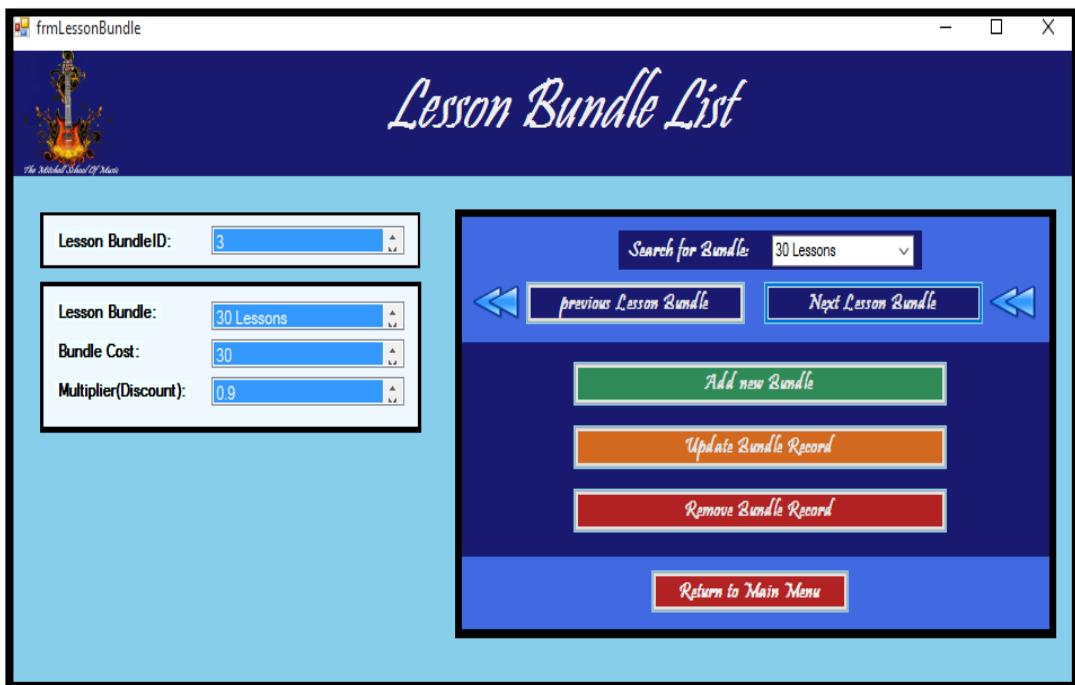
previous Lesson Bundle Next Lesson Bundle

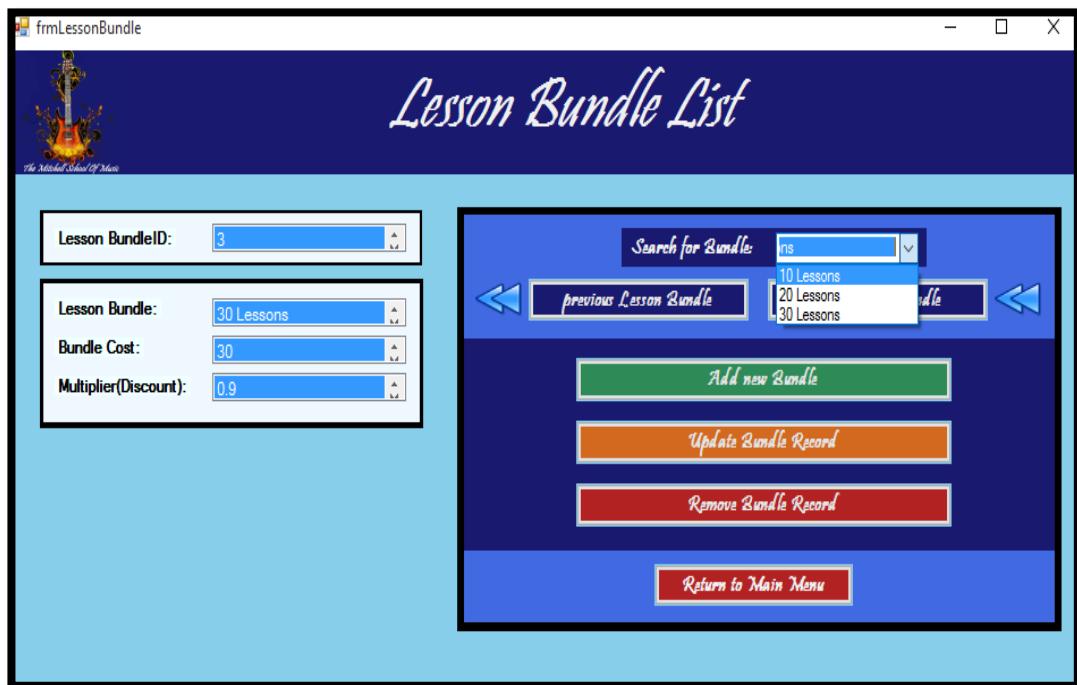
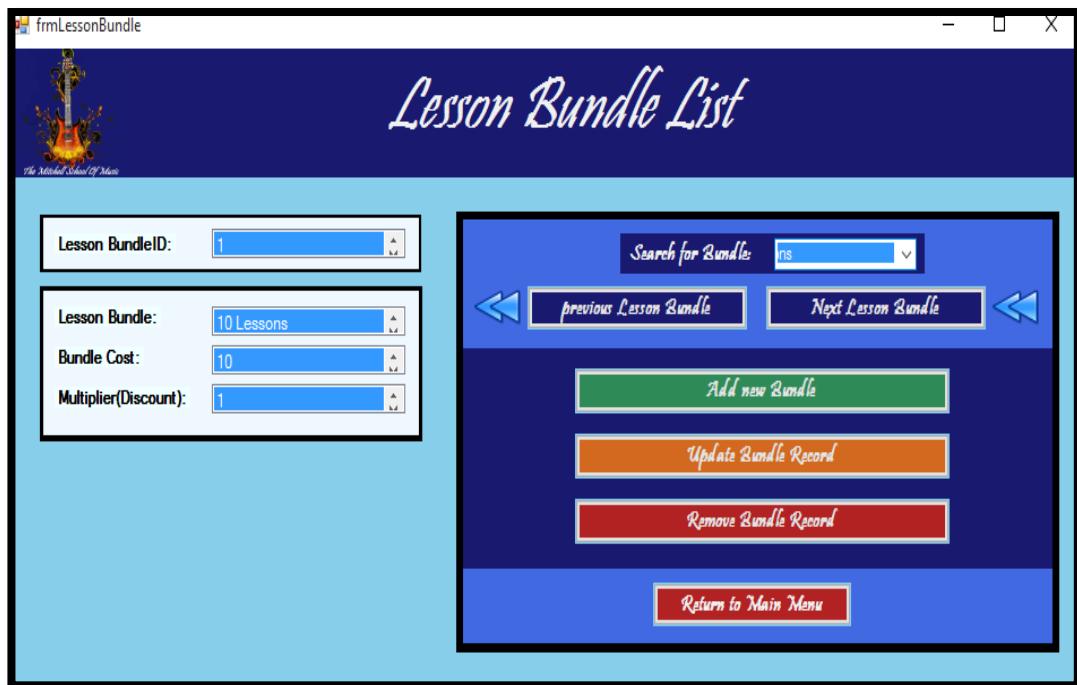
Add new Bundle

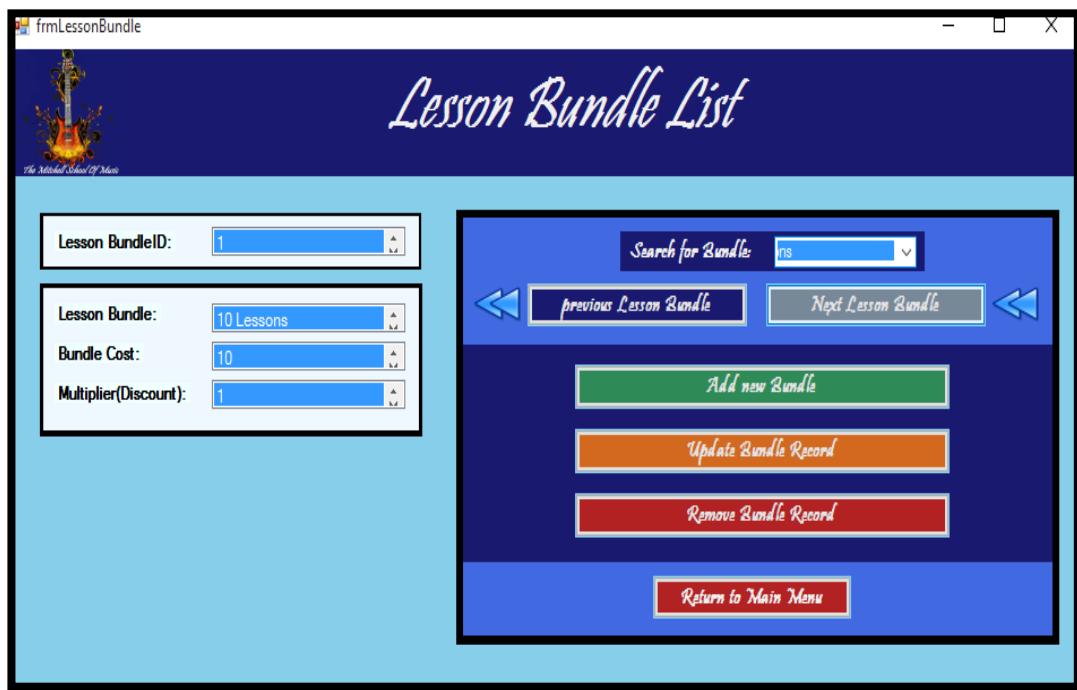
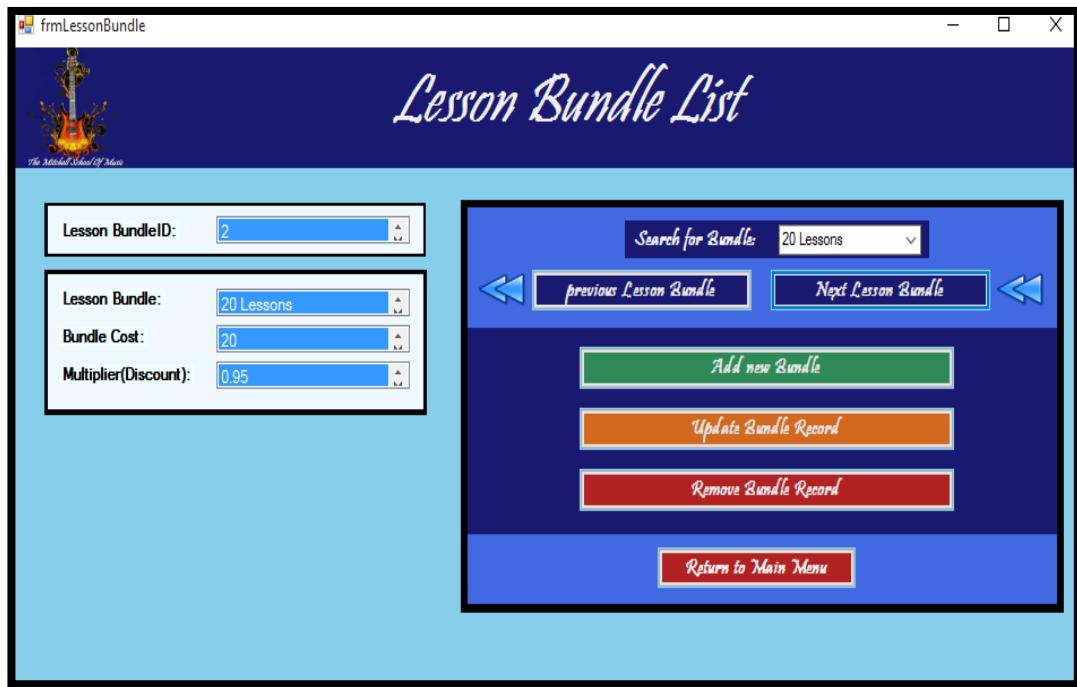
Update Bundle Record

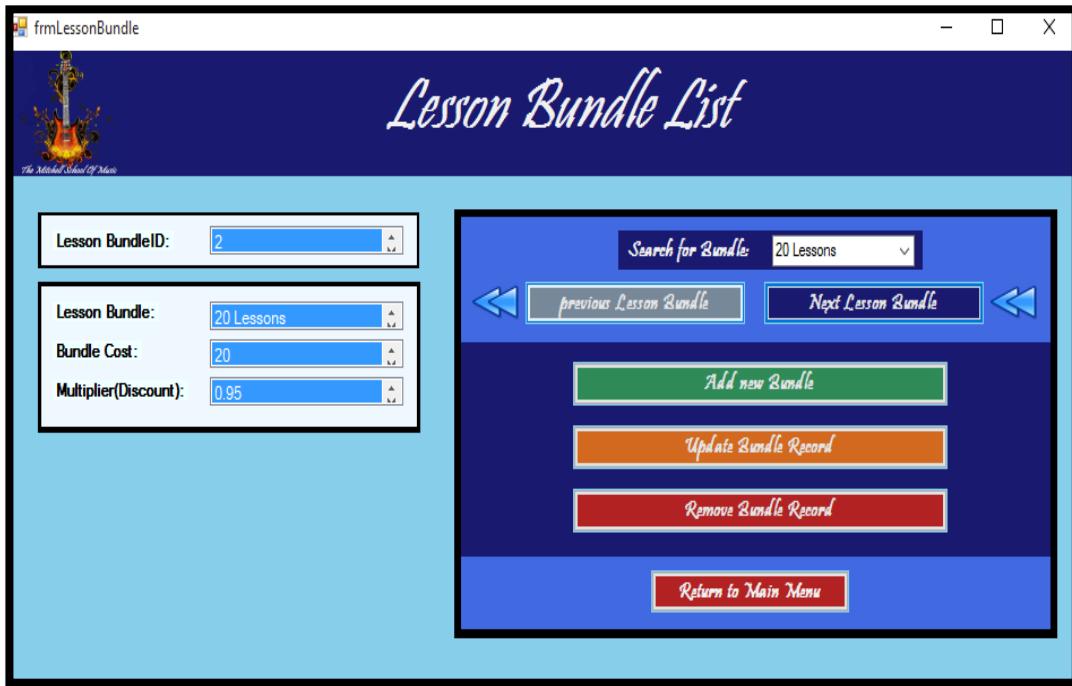
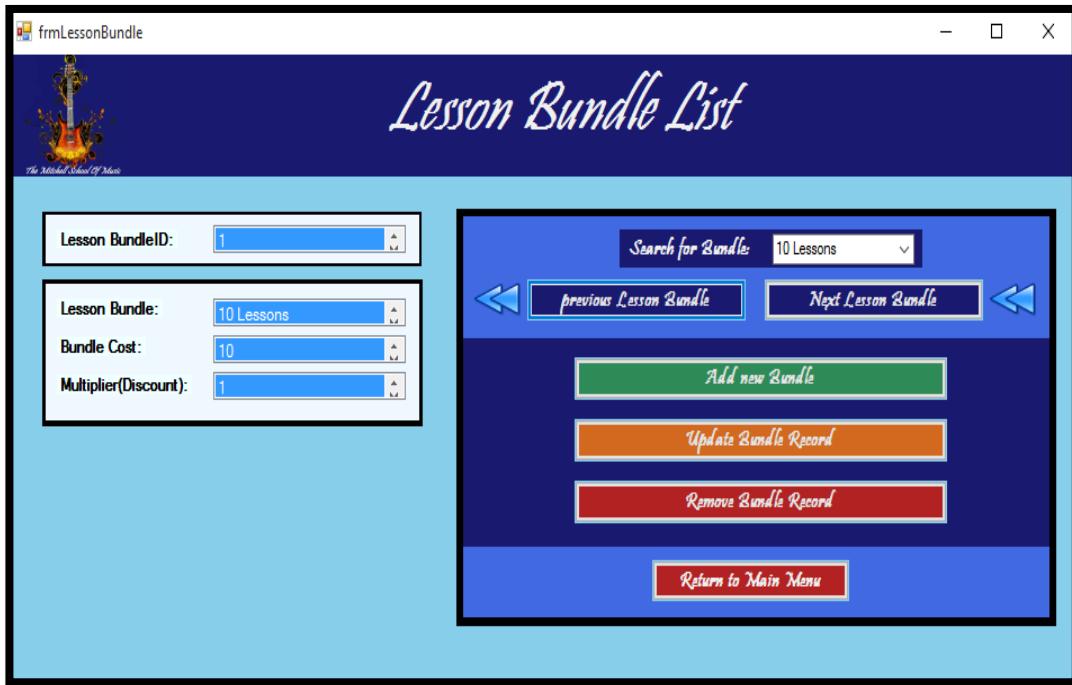
Remove Bundle Record

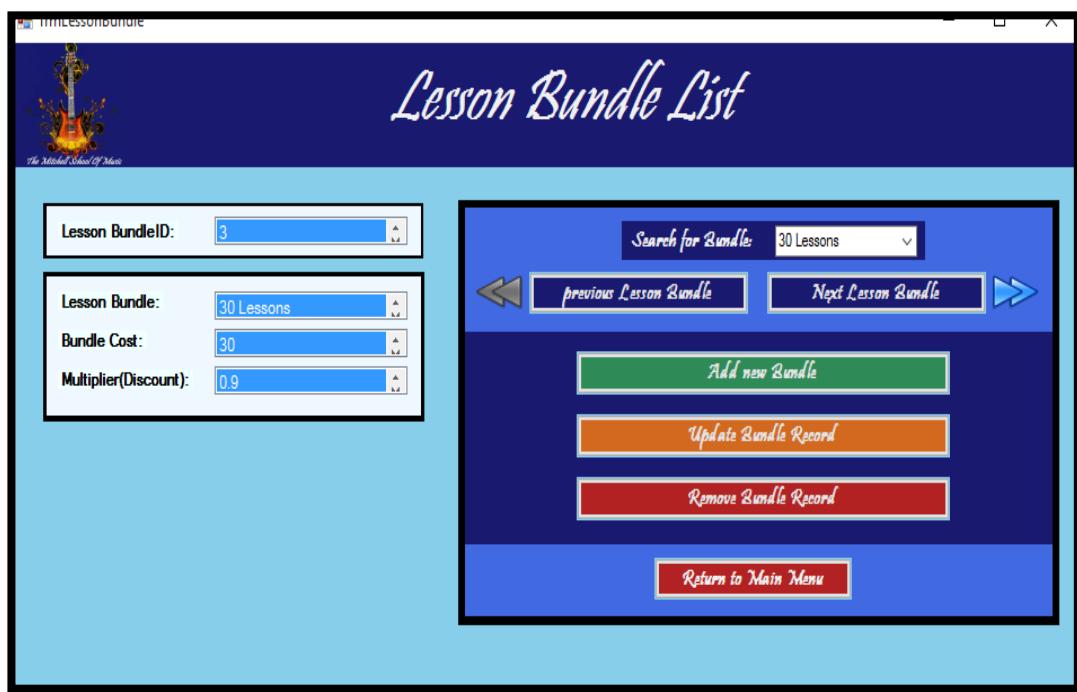
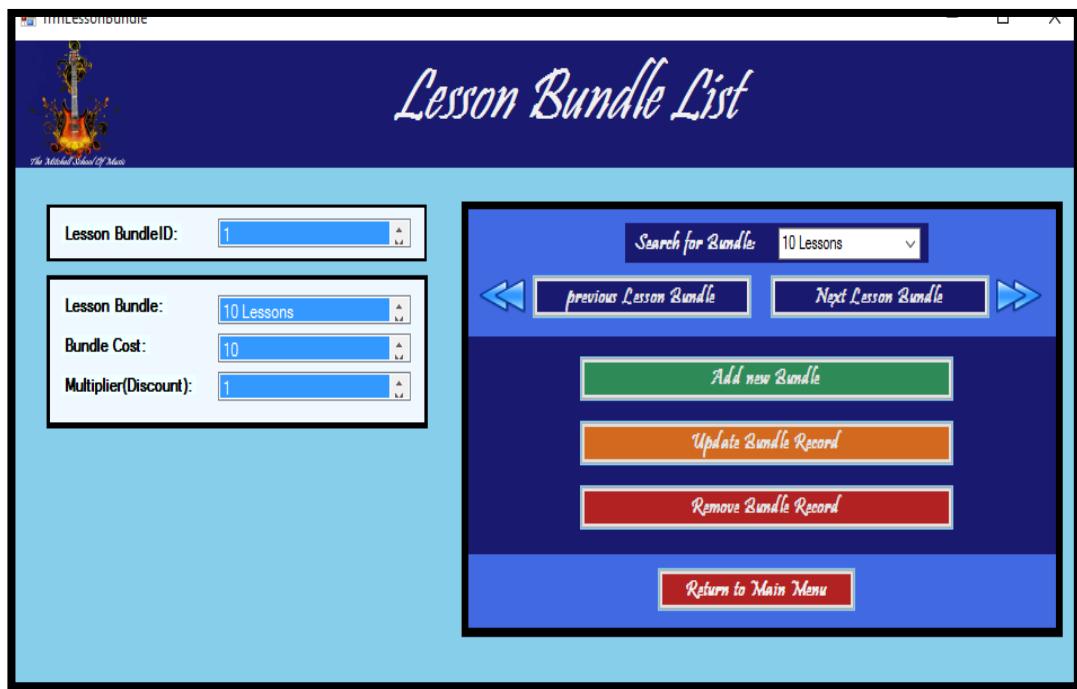
Return to Main Menu

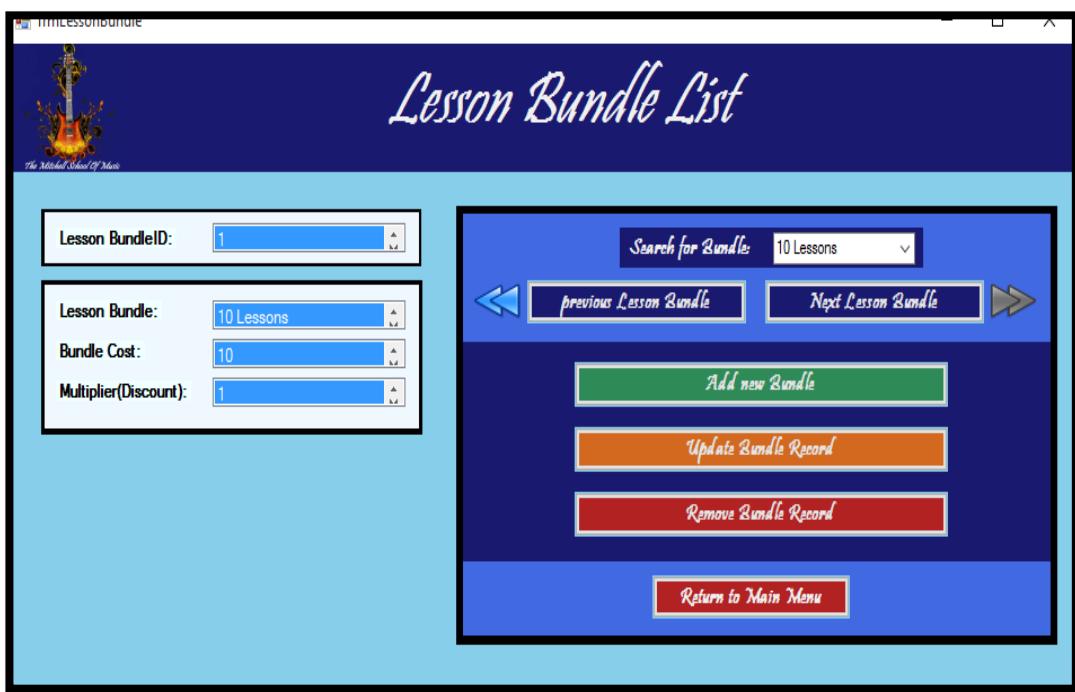
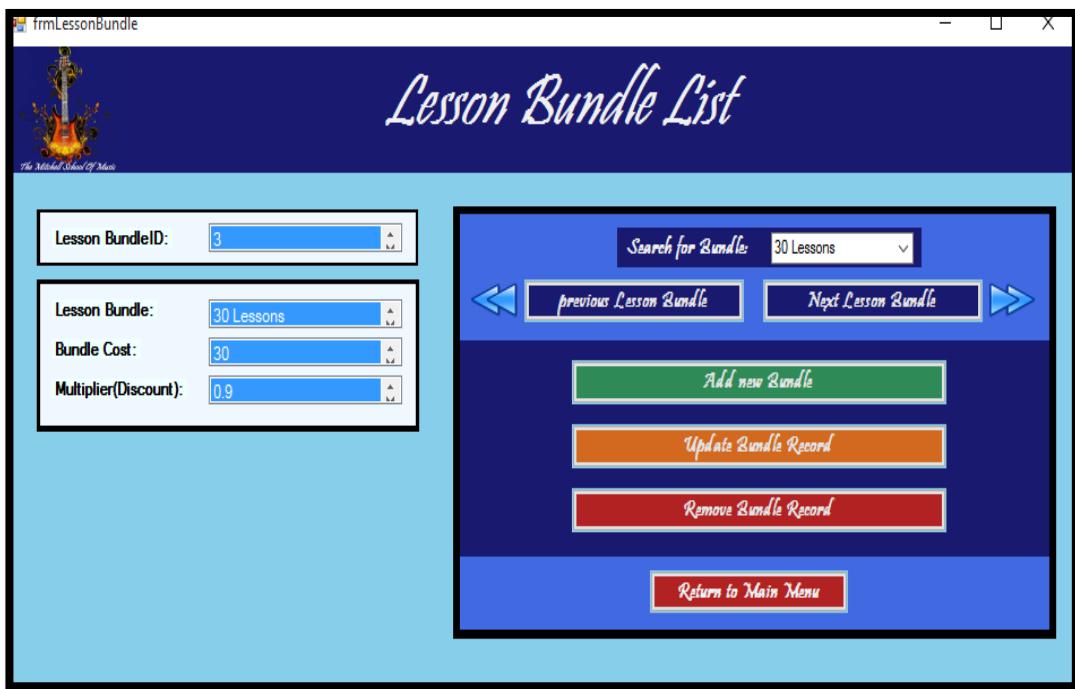
2.8[A][1] Screenshot C**2.8[A][1] Screenshot D**

*Process [B]) Record Navigation***2.8[B][1] – Bundle Search****2.8[B][1] Screenshot A****2.8[B][1] Screenshot B**

2.8[B][2] – Next Bundle**2.8[B][2] Screenshot A****2.8[B][2] Screenshot B**

2.8[B][3] – Previous Bundle**2.8[B][3] Screenshot A****2.8[B][3] Screenshot B**

2.8[B][4] – First Bundle**2.8[B][4] Screenshot A****2.8[B][4] Screenshot B**

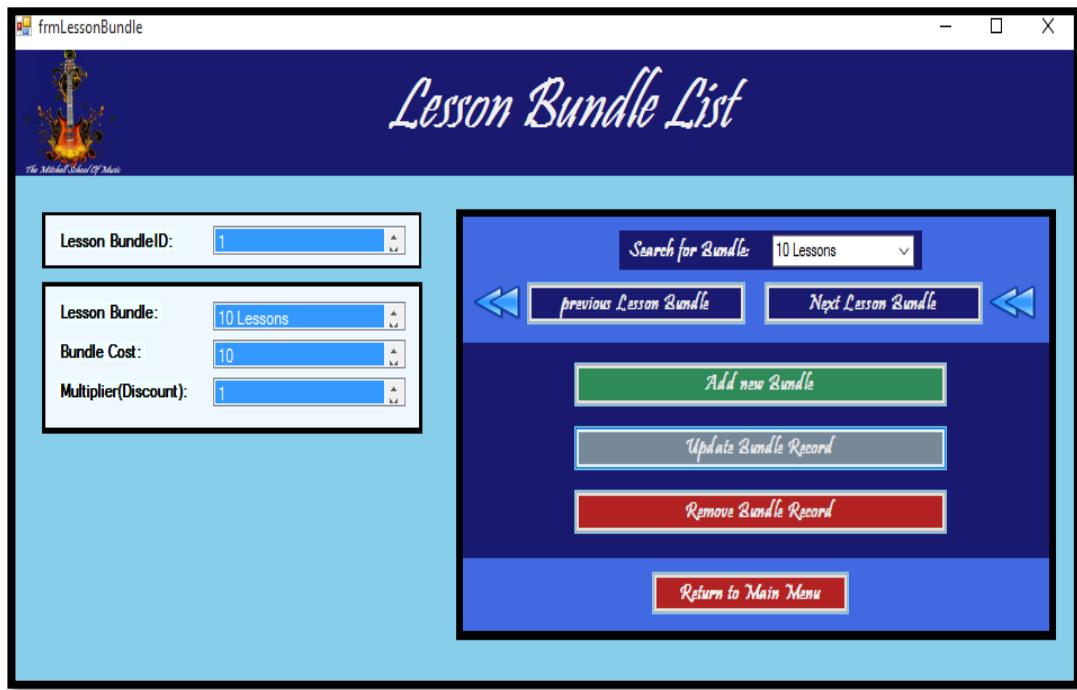
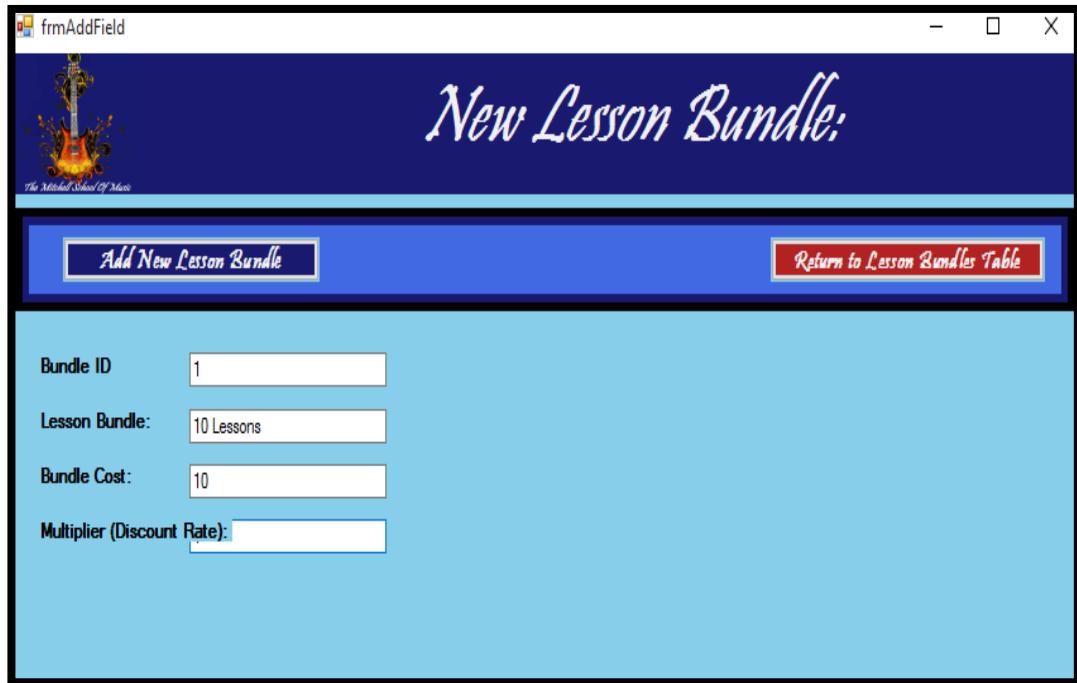
2.8[B][5] – Last Bundle**2.8[B][5] Screenshot A****2.8[B][5] Screenshot B**

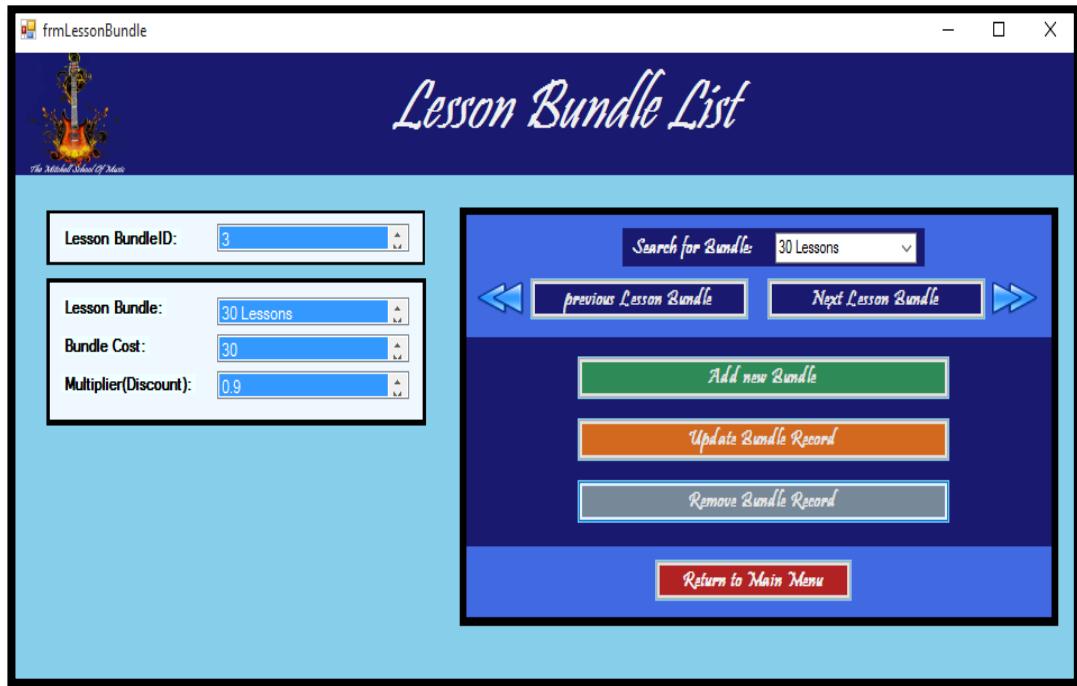
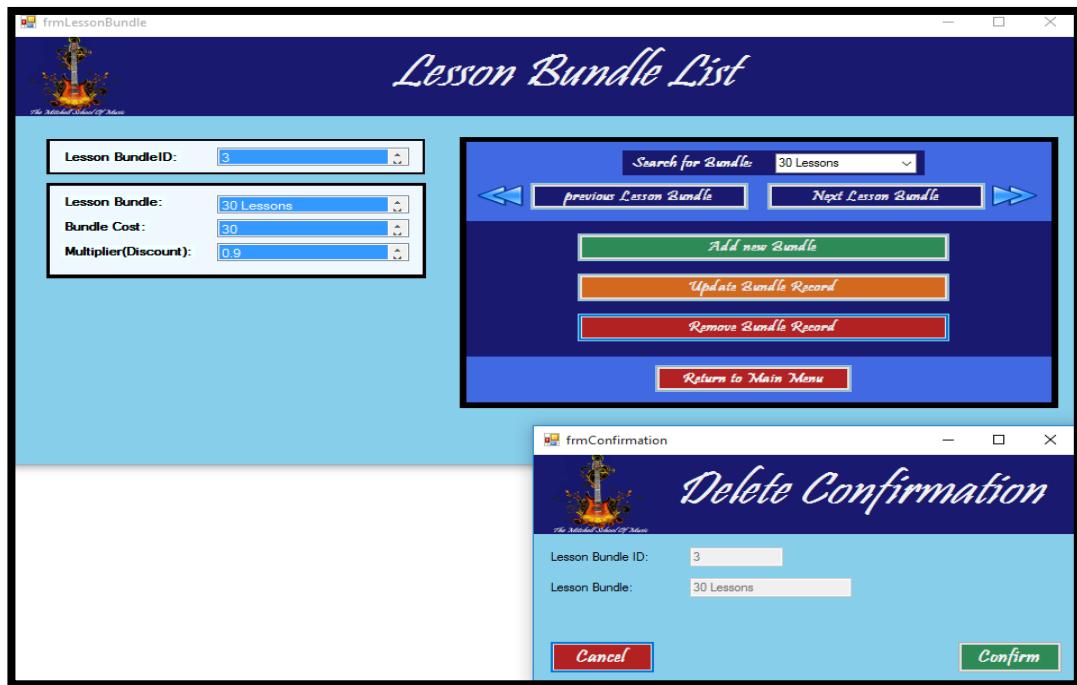
*Process [C]) Form Navigation**2.8[C][1] – Add New Bundle Record**2.8[C][1] Screenshot A*

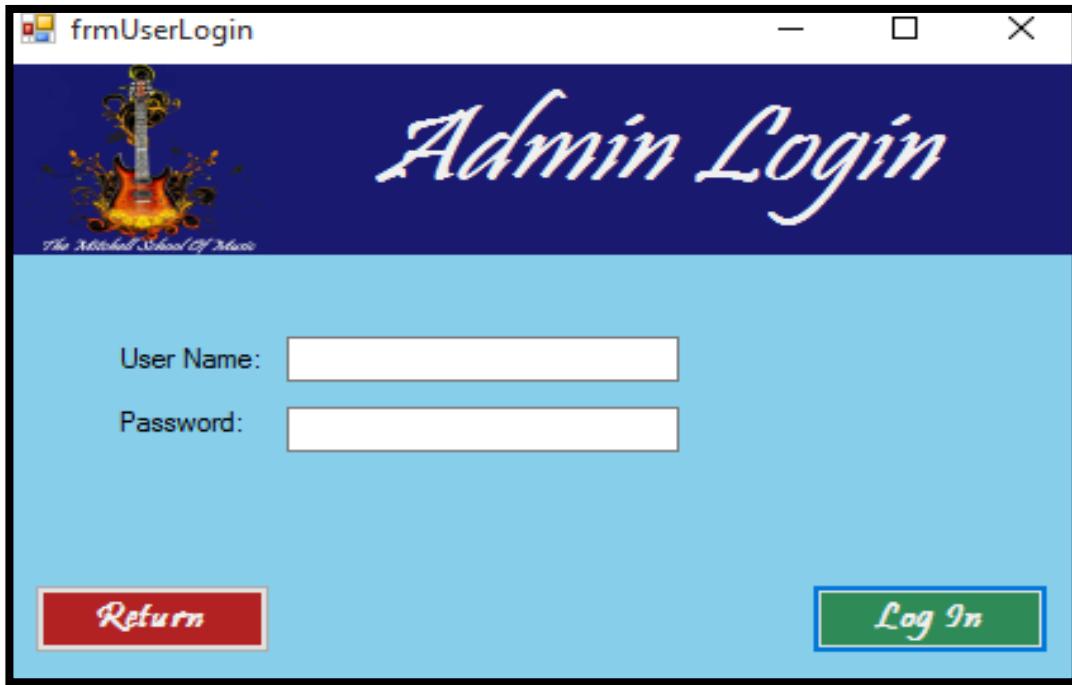
The screenshot shows a Windows application window titled "frmLessonBundle". The title bar includes standard window controls. The main title "Lesson Bundle List" is displayed prominently in a large, decorative font. In the top left corner, there is a logo of a guitar and the text "The Mitchell School Of Music". On the left side of the interface, there is a vertical stack of four input fields: "Lesson BundleID" (containing "1"), "Lesson Bundle" (containing "10 Lessons"), "Bundle Cost" (containing "10"), and "Multiplier(Discount)" (containing "1"). To the right of these fields is a search bar labeled "Search for Bundle" with a dropdown menu showing "10 Lessons". Below the search bar are two navigation buttons: "Previous Lesson Bundle" and "Next Lesson Bundle", each flanked by double-headed arrows. Underneath these buttons are three action buttons: "Add new Bundle" (light blue), "Update Bundle Record" (orange), and "Remove Bundle Record" (red). At the bottom of the screen is a "Return to Main Menu" button.

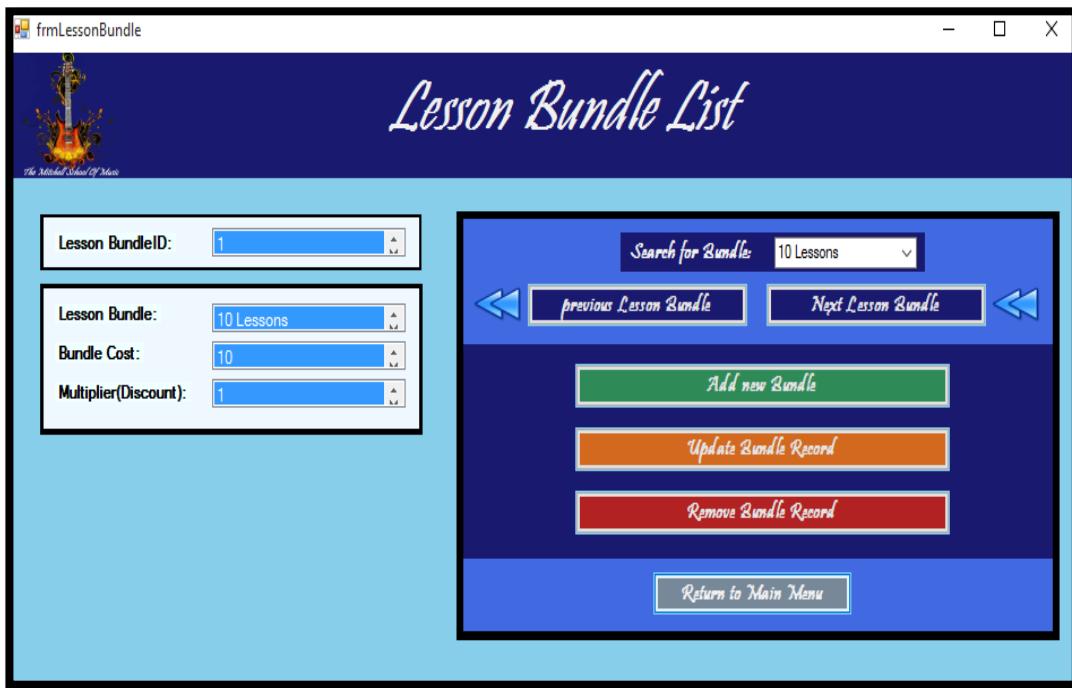
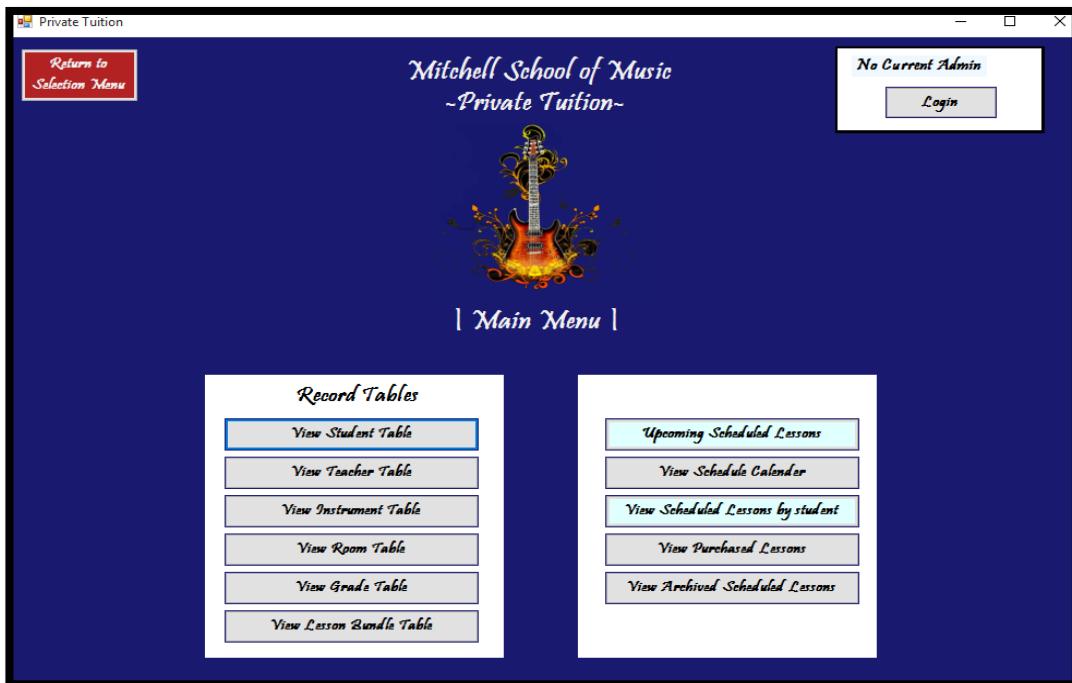
2.8[C][1] Screenshot B

The screenshot shows a Windows application window titled "frmAddField". The title bar includes standard window controls. The main title "New Lesson Bundle;" is displayed prominently in a large, decorative font. In the top left corner, there is a logo of a guitar and the text "The Mitchell School Of Music". At the top of the interface are two buttons: "Add New Lesson Bundle" on the left and "Return to Lesson Bundles Table" on the right. Below these buttons is a horizontal line separating the header from the input area. The input area contains three text input fields: "Lesson Bundle:" (empty), "Bundle Cost:" (empty), and "Multiplier (Discount Rate):" (empty). The entire application has a light blue background.

2.8[C][2] – Update Bundle Record**2.8[C][2] Screenshot A****2.8[C][2] Screenshot B**

2.8[C][3] – Delete Bundle Record [User Logged In]**2.8[C][3] Screenshot A****2.8[C][3] Screenshot B**

2.8[C][4] – Delete Bundle Record [No User Logged In]**2.8[C][4] Screenshot A****2.8[C][4] Screenshot B**

2.8[C][5] – Return To Main Menu**2.8[C][5] Screenshot A****2.8[C][5] Screenshot B**

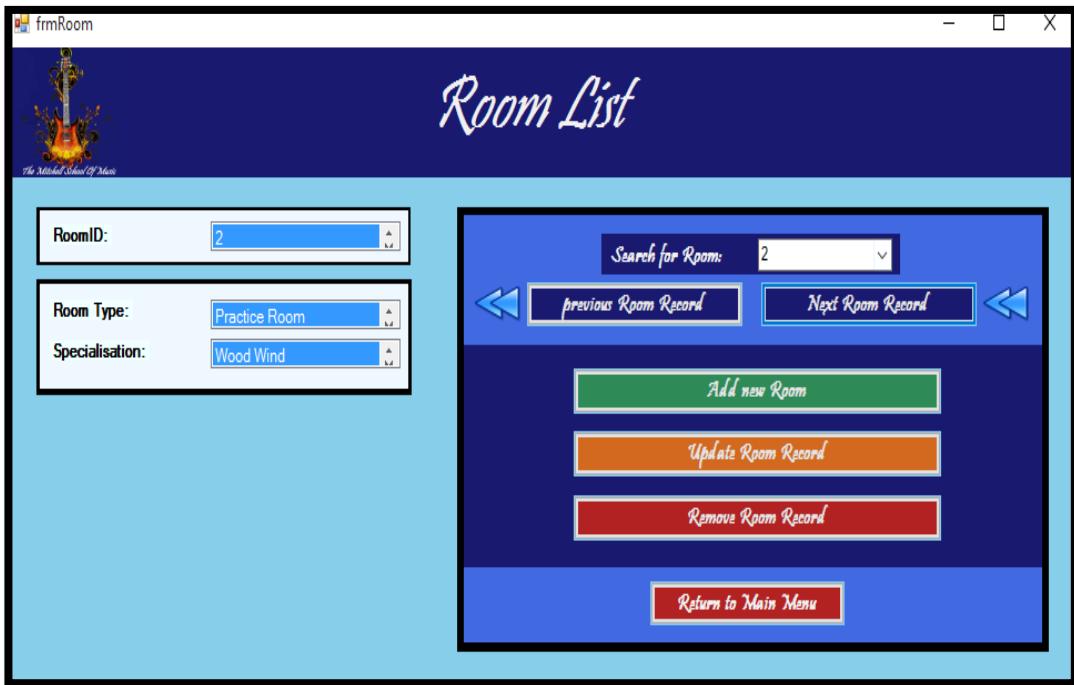
2.9) Room Form***Process [A] Information Display******2.9[A][1] – Room Information******2.9[A][1] Screenshot A***

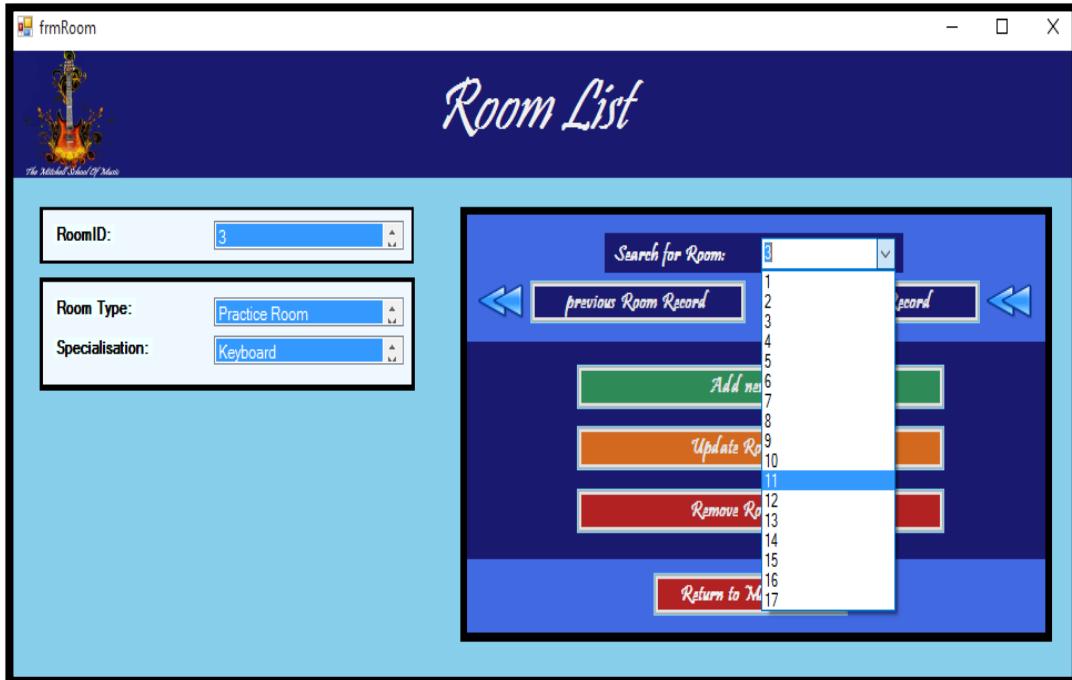
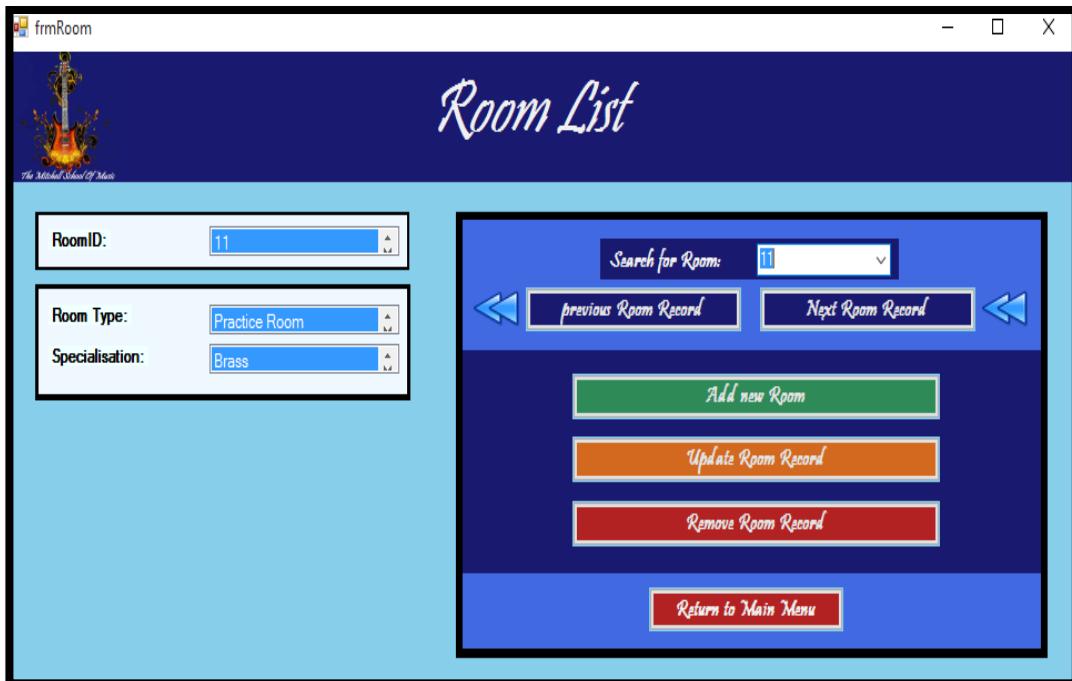
	RoomID	Room_Type	Specialisation
1	Practice Room	String	...
2	Practice Room	Wood Wind	...
3	Practice Room	Keyboard	...
4	Practice Room	Precussion	...
5	Practice Room	Brass	...
6	Practice Room	Vocal	...
7	Practice Room	String	...
8	Practice Room	Keyboard	...
9	Practice Room	Precussion	...
10	Practice Room	Vocal	...
11	Practice Room	Brass	...
12	Practice Room	Wood Wind	...
13	Classroom	String & Precus...	...
14	Classroom	Woodwind & B...	...
15	Classroom	Keyboard	...
16	Classroom	Vocal	...
17	Hall	Orchestra	...
0	NULL	NULL	NULL

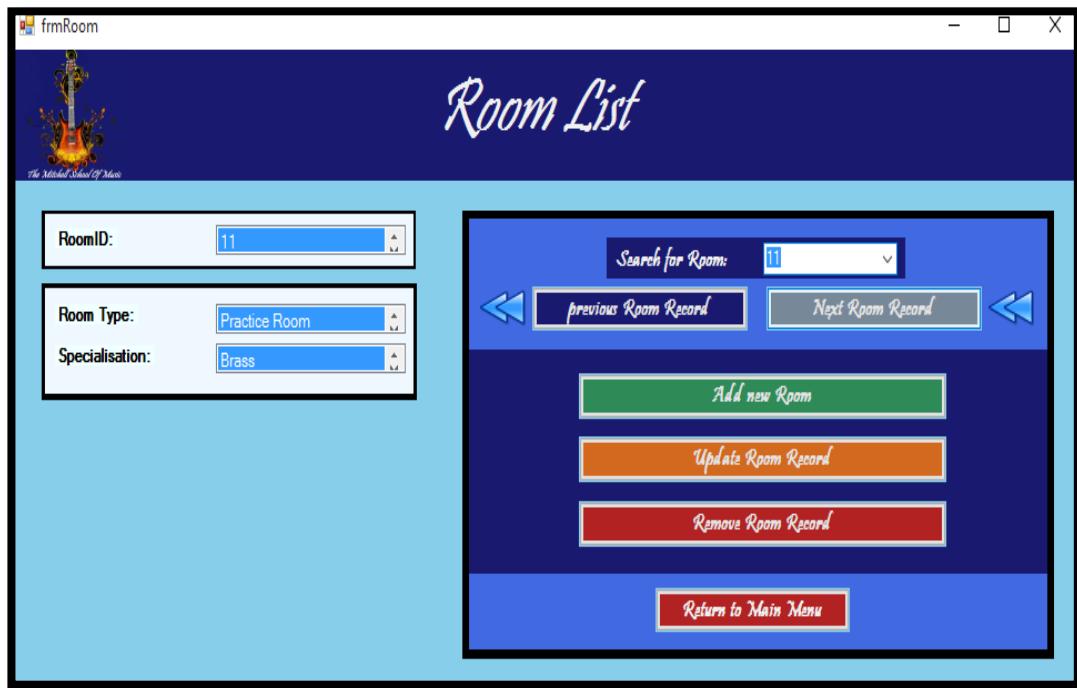
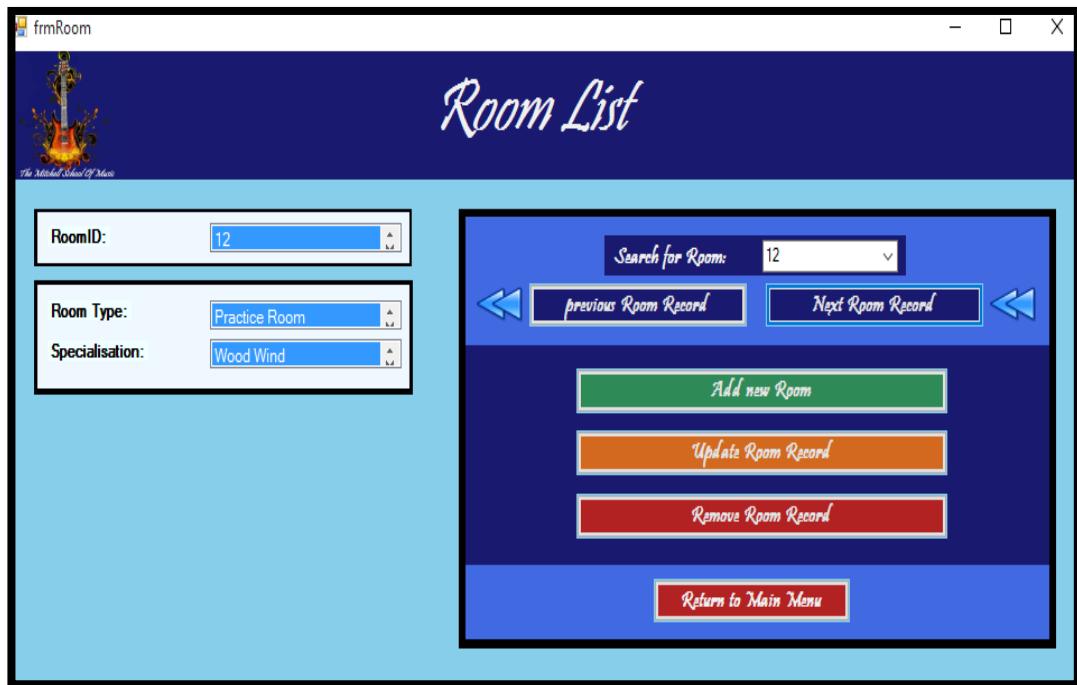
2.9[A][1] Screenshot B

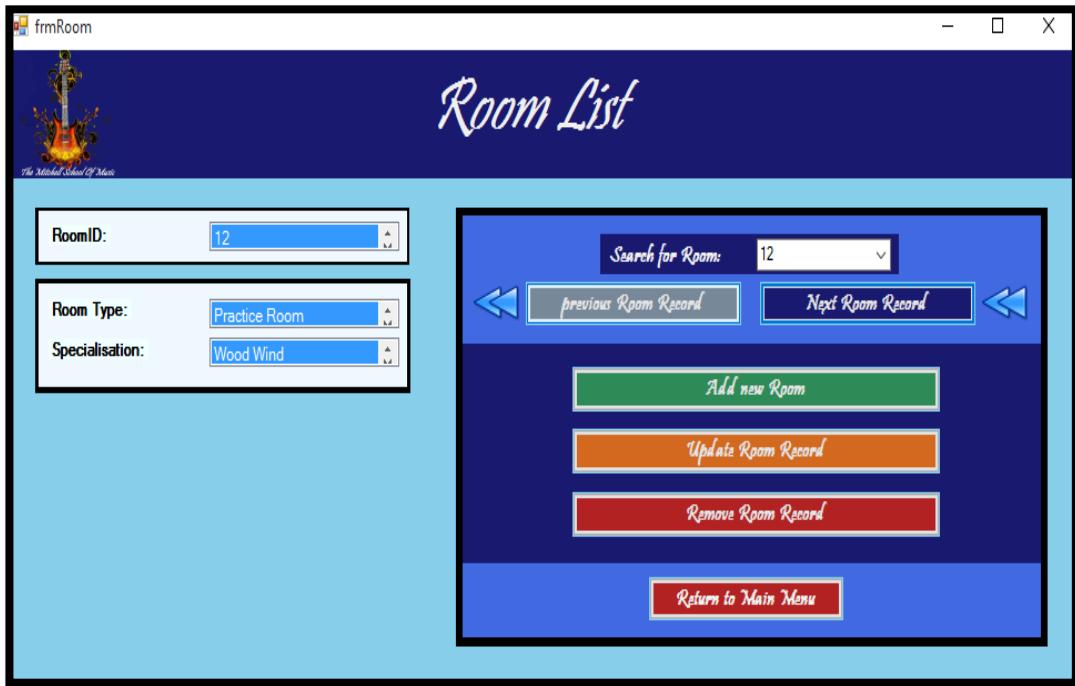
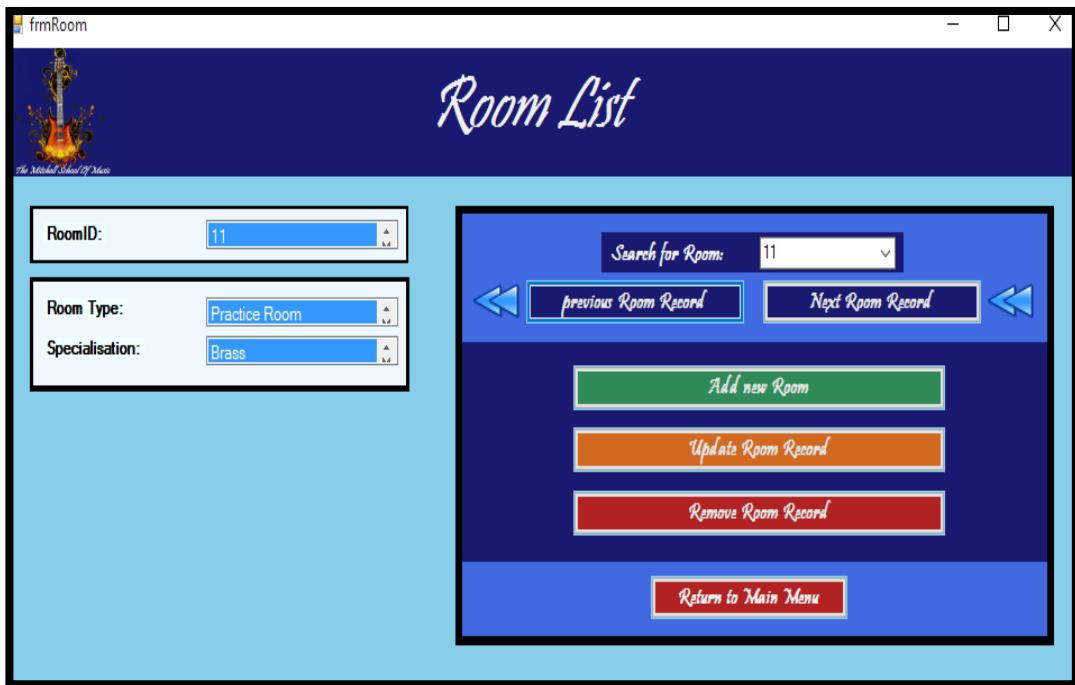
The application window title is "frmRoom". The main title bar says "Room List". The interface includes:

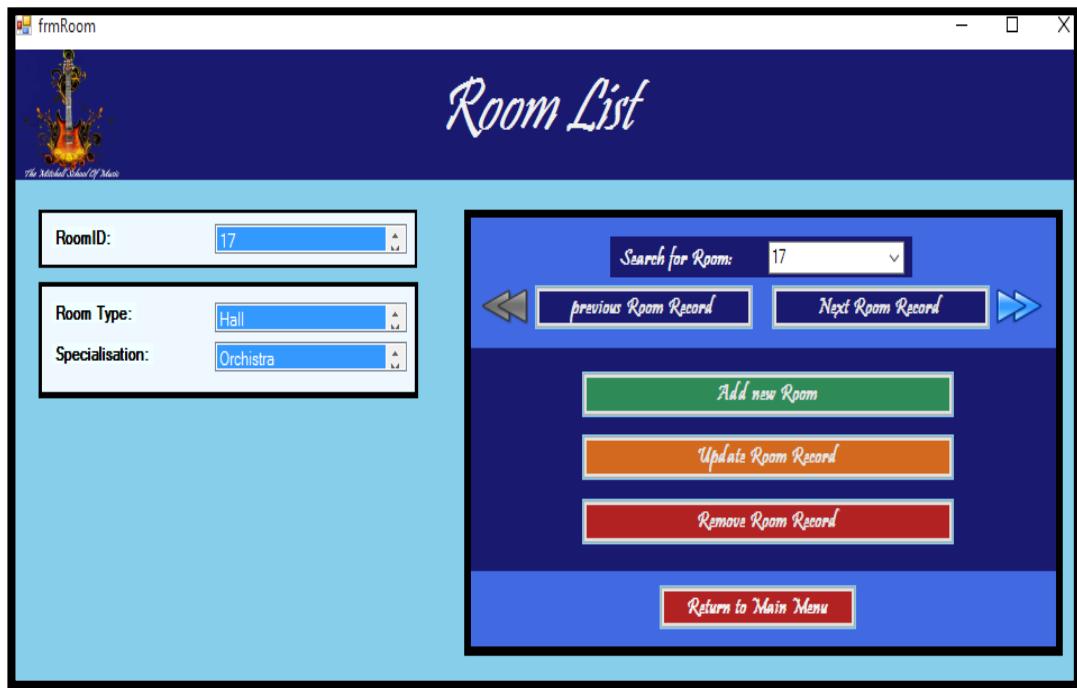
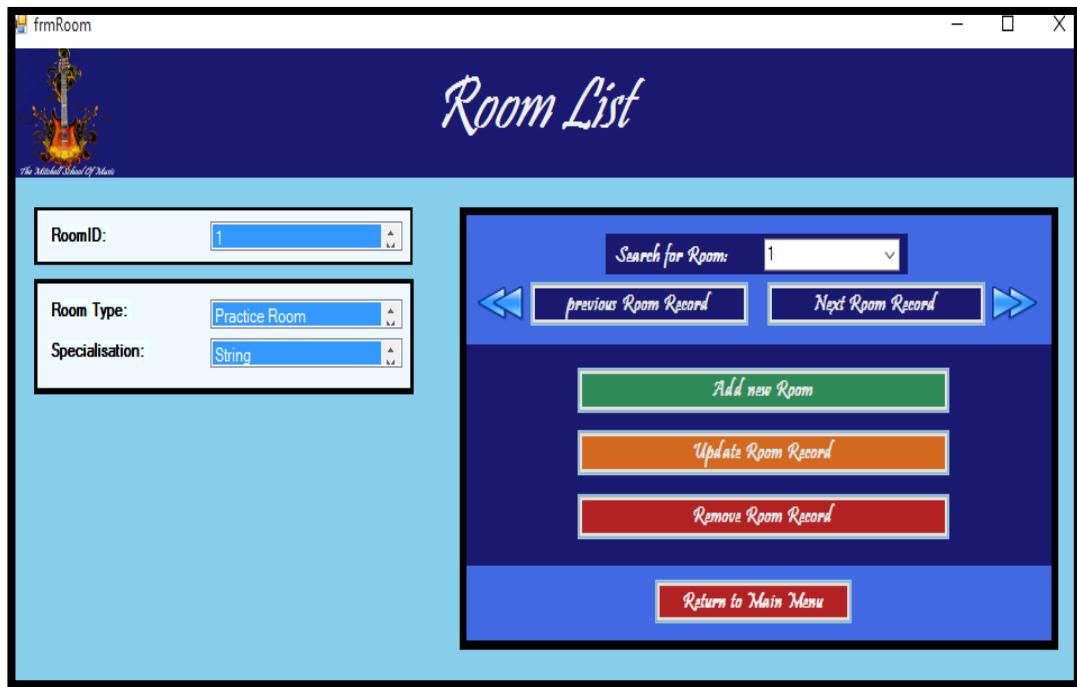
- A logo for "The Metal School Of Music" featuring a guitar.
- Search and navigation controls: "Search for Room:" with value "1", "previous Room Record", "Next Room Record", and double arrow buttons.
- Action buttons: "Add new Room" (green), "Update Room Record" (orange), and "Remove Room Record" (red).
- A "Return to Main Menu" button at the bottom.
- Input fields: "RoomID:" with value "1", "Room Type:" with value "Practice Room", and "Specialisation:" with value "String".

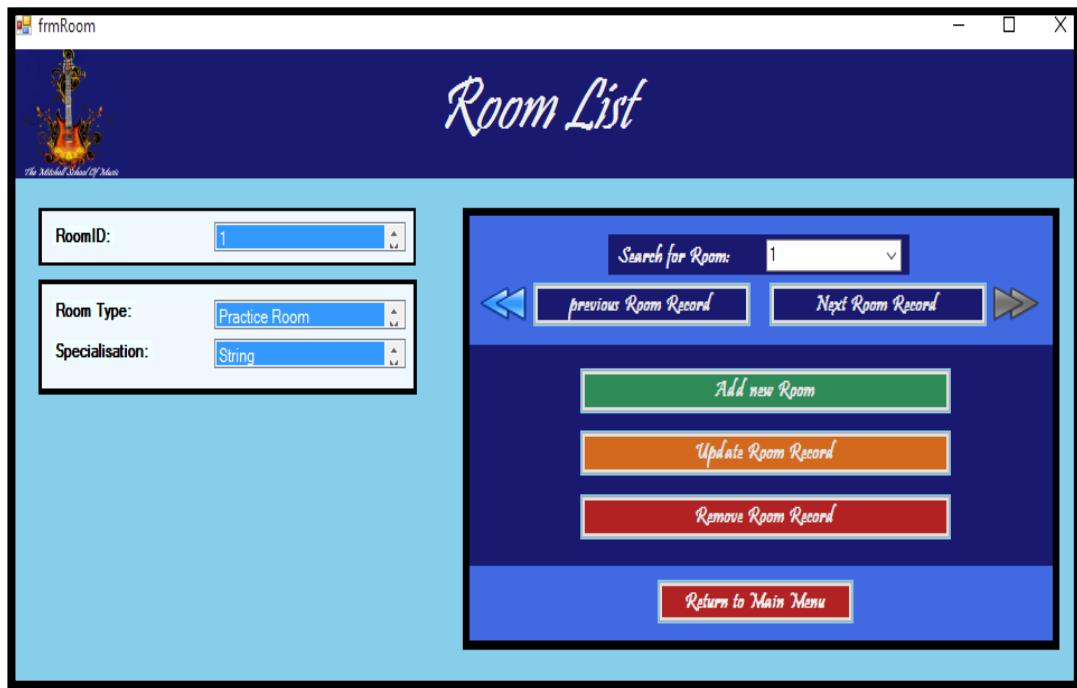
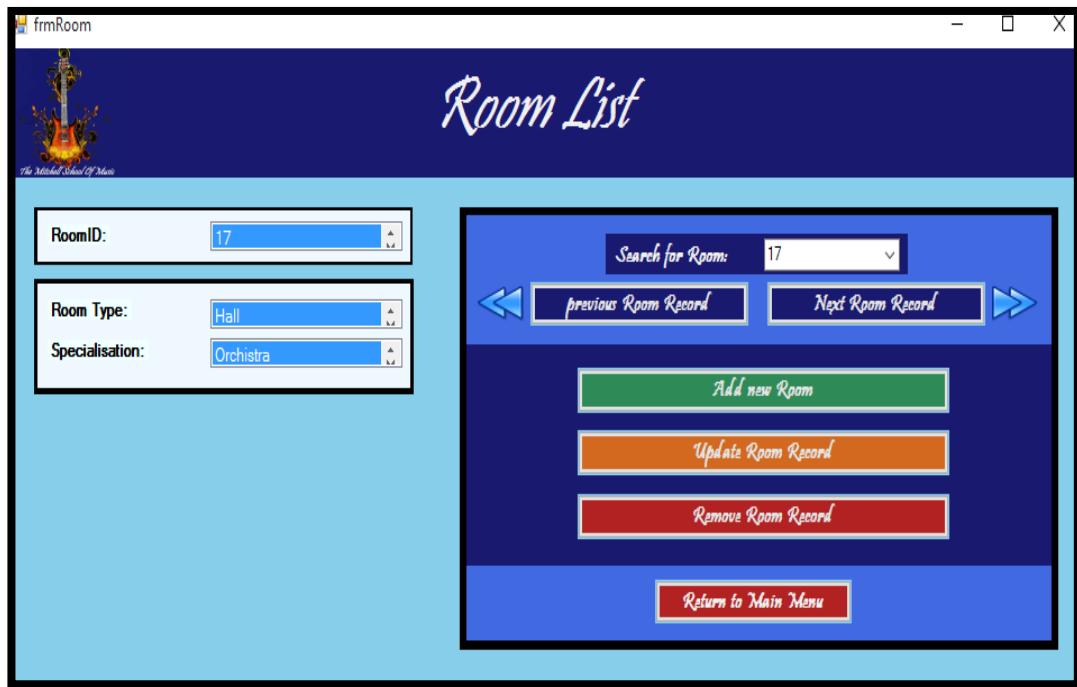
2.9[A][1] Screenshot C**2.9[A][1] Screenshot D**

*Process [B]) Record Navigation***2.9[B][1] – Room Search****2.9[B][1] Screenshot A****2.9[B][1] Screenshot B**

2.9[B][2] – Next Room**2.9[B][2] Screenshot A****2.9[B][2] Screenshot B**

2.9[B][3] – Previous Room**2.9[B][3] Screenshot A****2.9[B][3] Screenshot B**

2.9[B][4] – First Room**2.9[B][4] Screenshot A****2.9[B][4] Screenshot B**

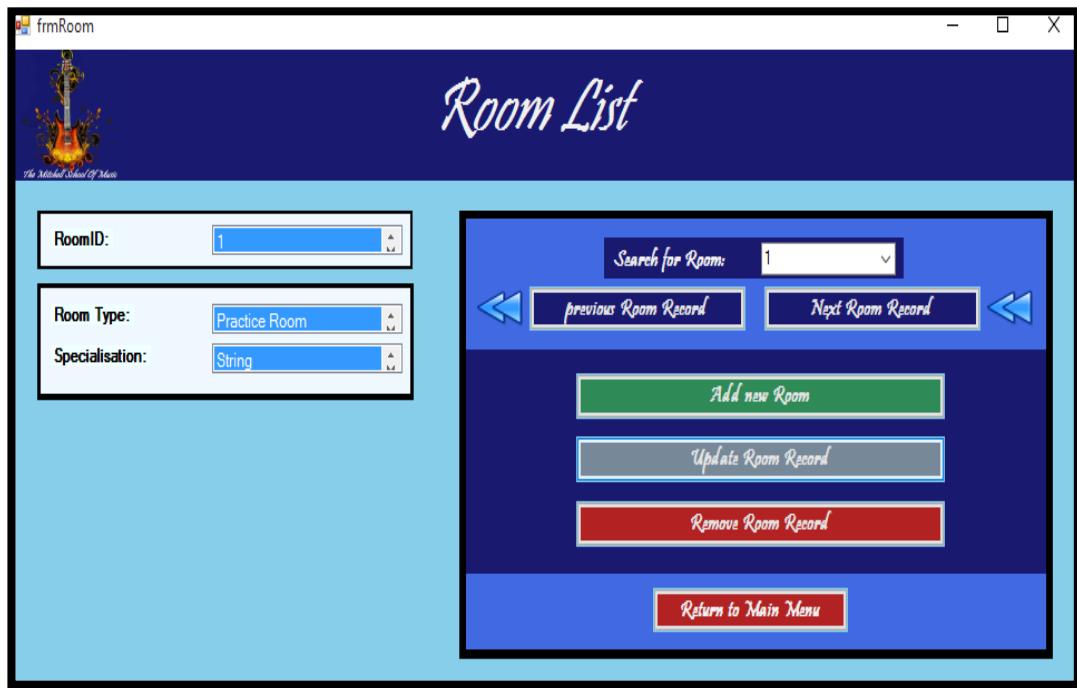
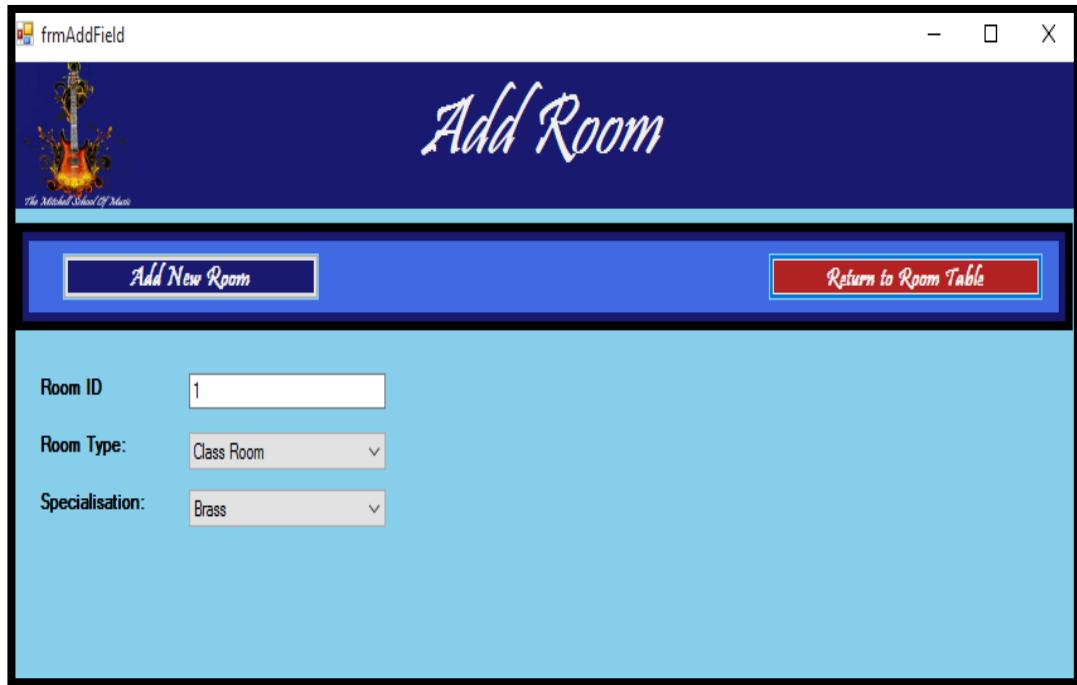
2.9[B][5] – Last Room**2.9[B][5] Screenshot A****2.9[B][5] Screenshot B**

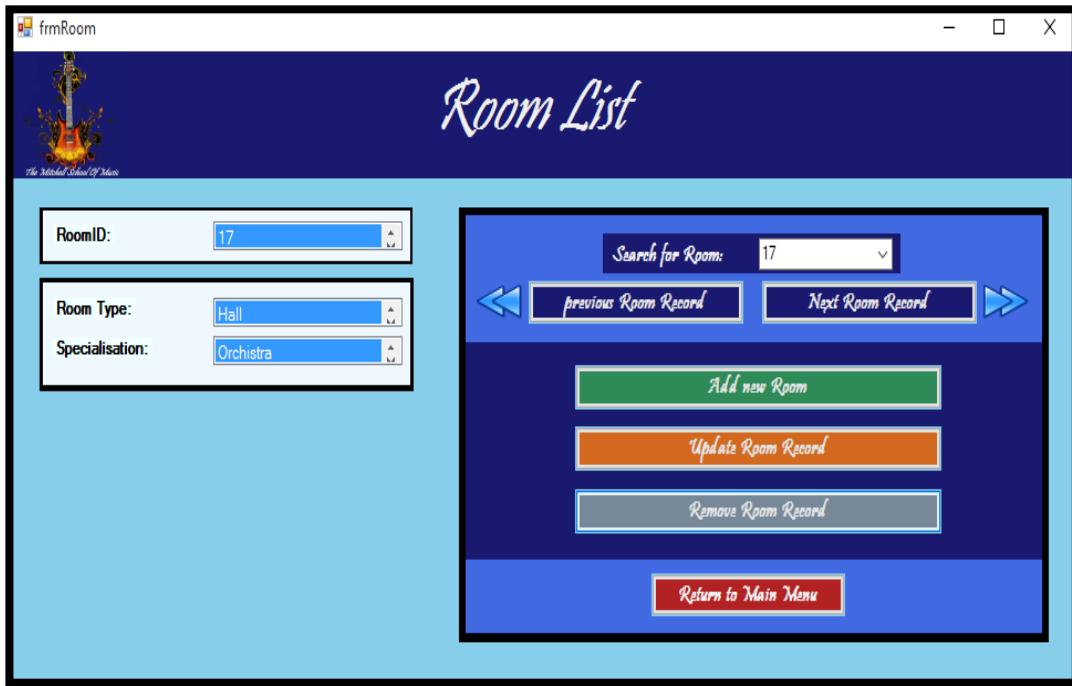
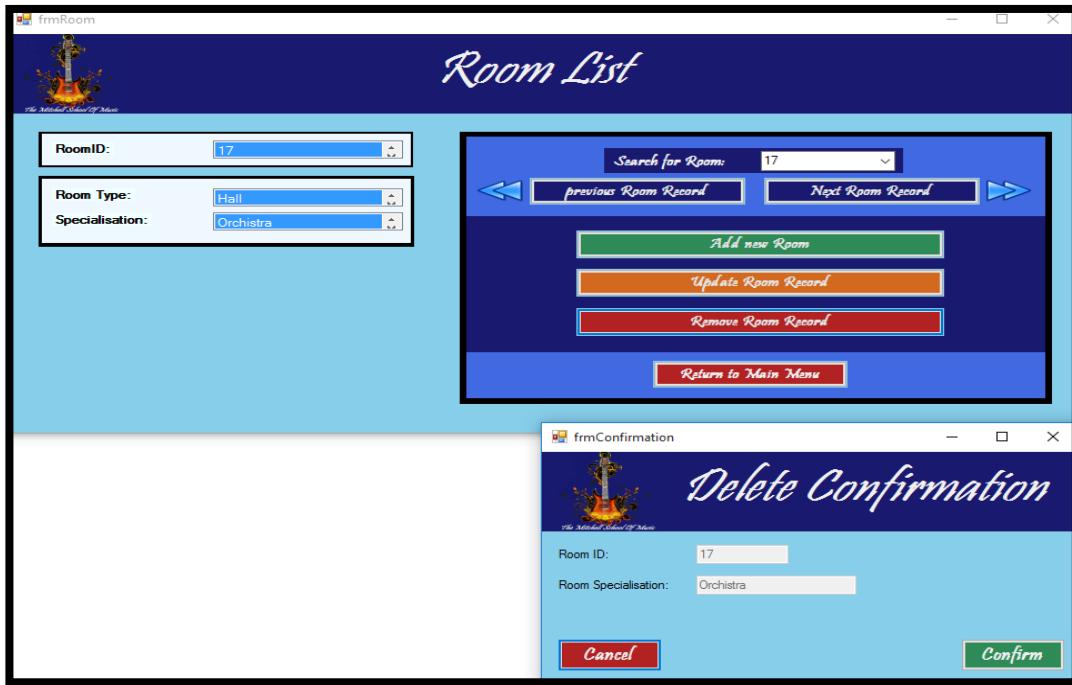
*Process [C]) Form Navigation***2.9[C][1] – Add New Room Record****2.9[C][1] Screenshot A**

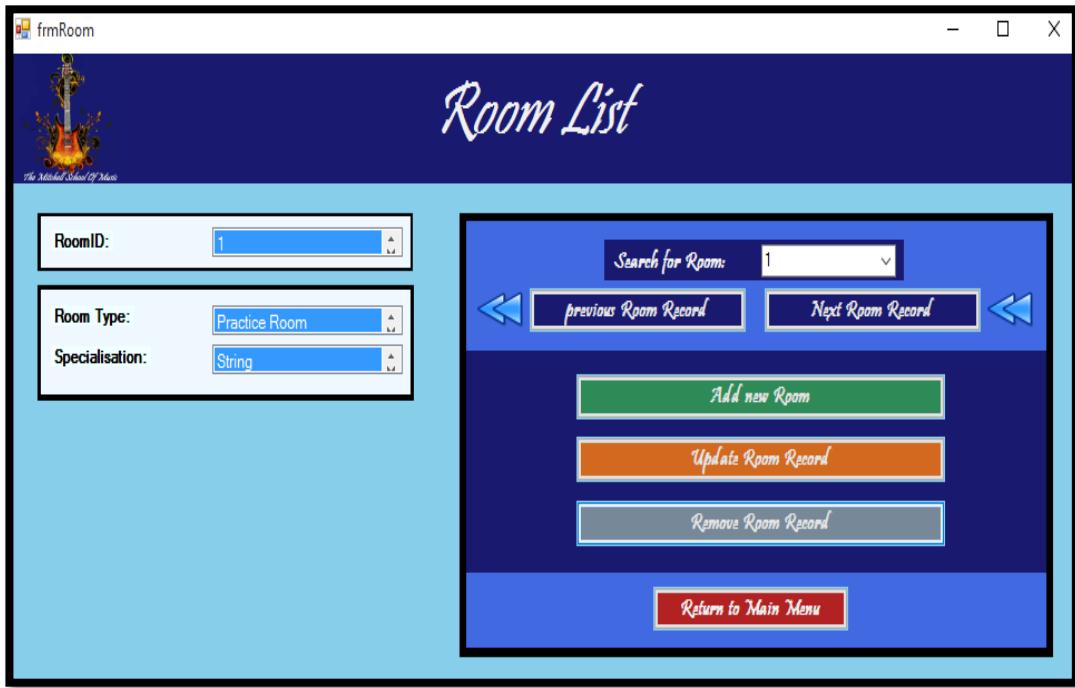
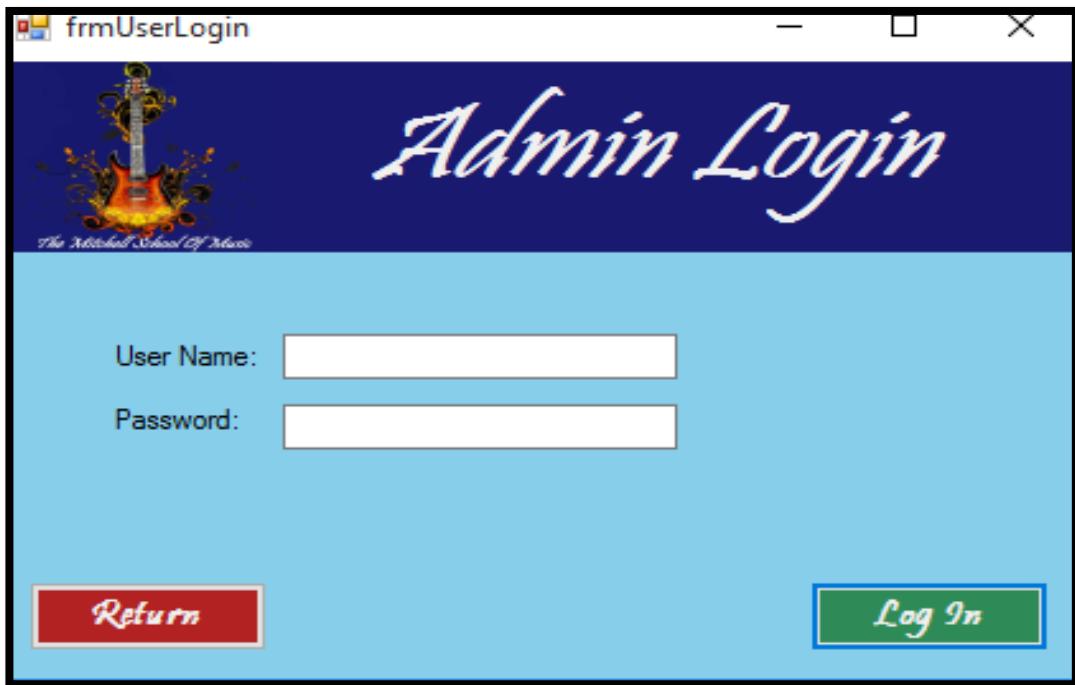
The screenshot shows the 'Room List' window (frmRoom). At the top left is a logo for 'The Mitchell School Of Music' featuring a stylized guitar. The main title 'Room List' is centered in a large, decorative font. On the left side, there are three input fields: 'RoomID' (containing '11'), 'Room Type' (containing 'Practice Room'), and 'Specialisation' (containing 'Brass'). To the right of these fields is a search bar labeled 'Search for Room:' with the value '11'. Below the search bar are two blue double-headed arrows pointing left and right, flanking two buttons: 'previous Room Record' and 'Next Room Record'. Underneath these are three colored buttons: light blue ('Add new Room'), orange ('Update Room Record'), and red ('Remove Room Record'). At the bottom is a blue button labeled 'Return to Main Menu'.

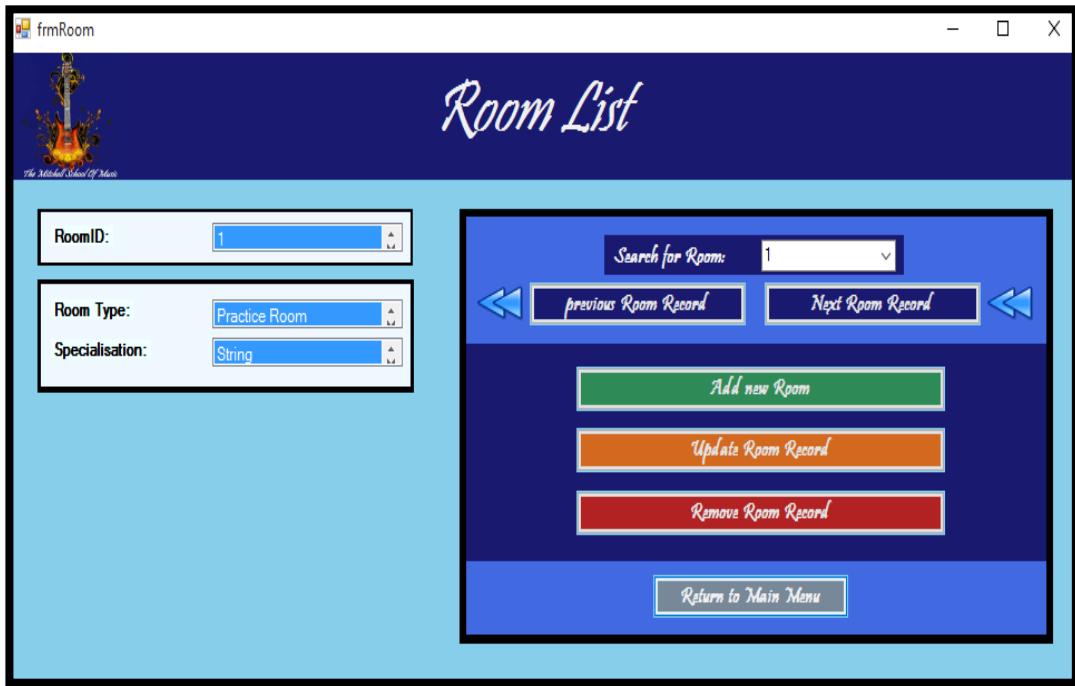
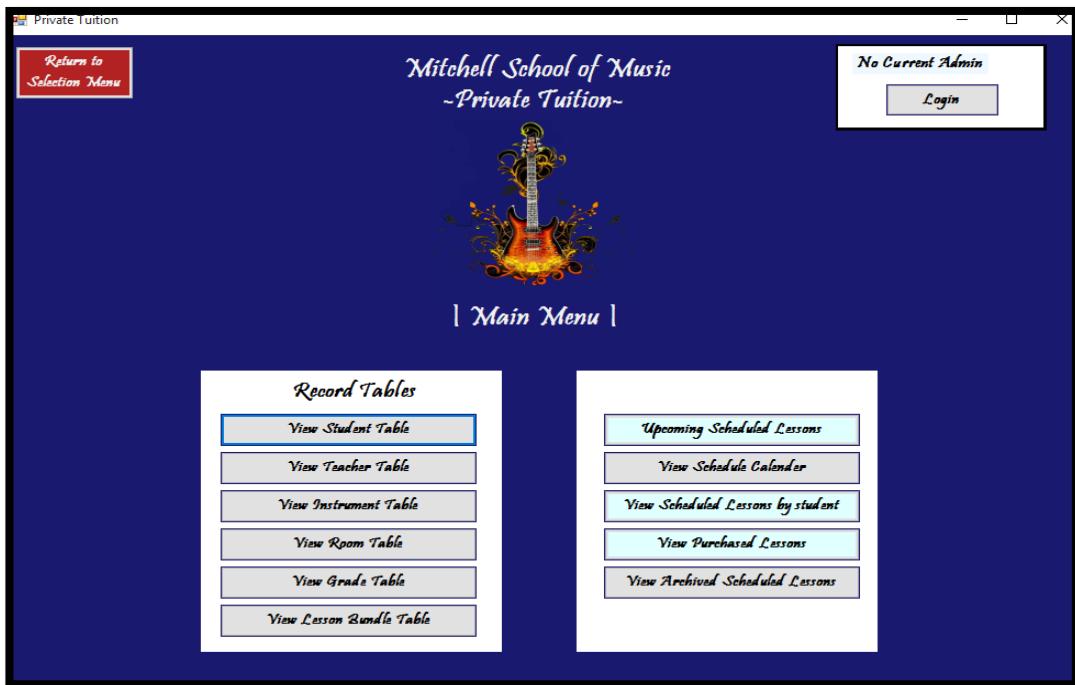
2.9[C][1] Screenshot B

The screenshot shows the 'Add Room' window (frmAddField). The title 'Add Room' is centered at the top in a large, decorative font. Below it are two buttons: 'Add New Room' on the left and 'Return to Room Table' on the right. The main area contains two dropdown menus: 'Room Type' (set to 'Class Room') and 'Specialisation' (set to 'Brass').

2.9[C][2] – Update Room Record**2.9[C][2] Screenshot A****2.9[C][2] Screenshot B**

2.9[C][3] – Delete Room Record [User Logged In]**2.9[C][3] Screenshot A****2.9[C][3] Screenshot B**

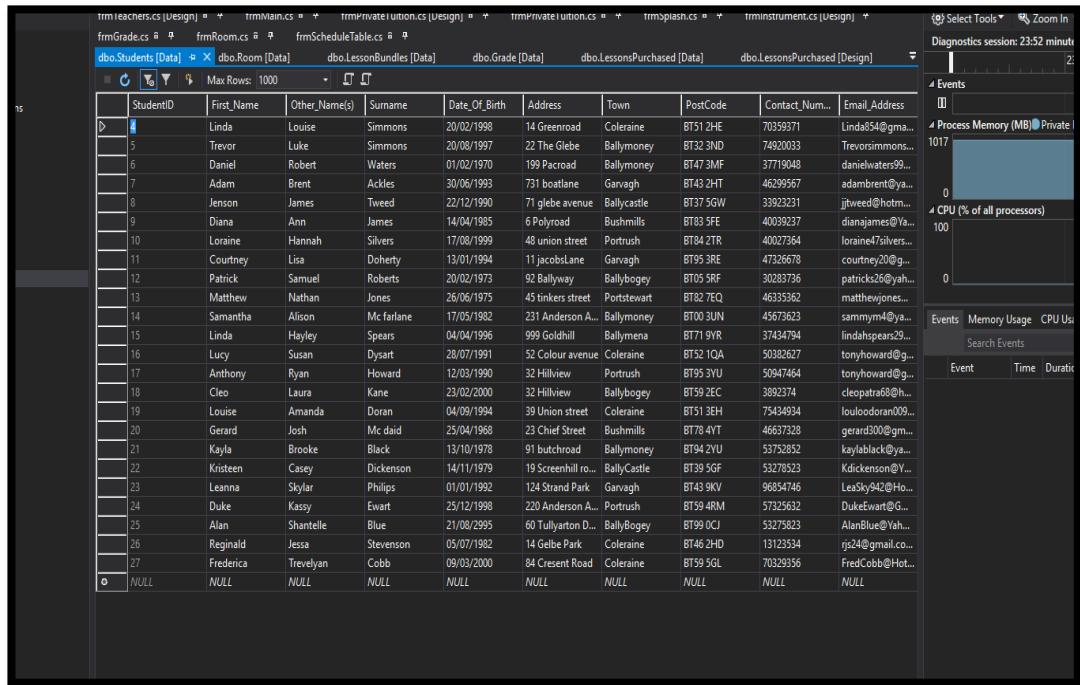
2.9[C][4] – Delete Room Record [No User Logged In]**2.9[C][4] Screenshot A****2.9[C][4] Screenshot B**

2.9[C][5] – Return To Main Menu**2.9[C][5] Screenshot A****2.9[C][5] Screenshot B**

2.10) Purchased Lessons Form
Process [A] Information Display

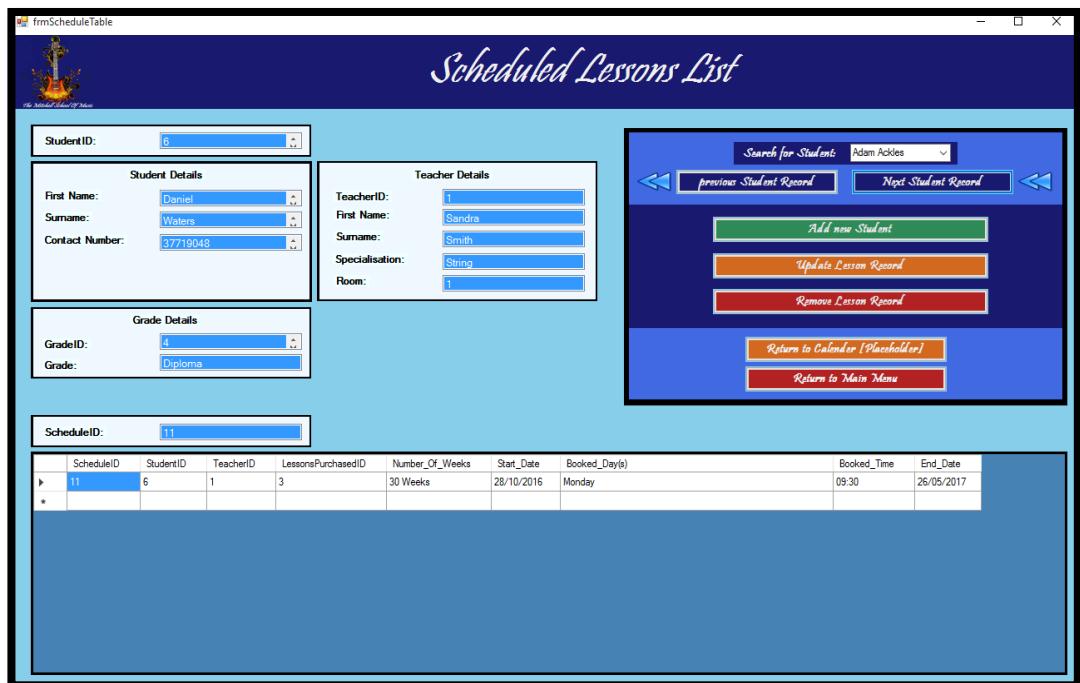
2.10[A][1] – Student Information

2.10[A][1] Screenshot A



	StudentID	First_Name	Other_Name(s)	Surname	DateOfBirth	Address	Town	PostCode	Contact_Number	Email_Address
1	1	Linda	Louise	Simmons	20/02/1998	14 Greenroad	Coleraine	BT51 2HE	70259371	Linda854@gma...
2	5	Trevor	Luke	Simmons	20/08/1997	22 The Glebe	Ballymoney	BT32 3ND	74920033	Trevor@simmons...
3	6	Daniel	Robert	Waters	01/02/1970	199 Parcroad	Ballymoney	BT47 3MF	37719048	danielwaters99...
4	7	Adam	Brent	Ackles	30/06/1993	731 boatlane	Ganagh	BT43 2HT	46299567	adambrent@ya...
5	8	Jenson	James	Tweed	22/12/1990	71 glebe avenue	Ballycastle	BT37 5GW	33923231	jrtweed@hotm...
6	9	Diana	Ann	James	14/04/1985	6 Polyroad	Bushmills	BT83 5FE	40039237	dianajames@Ya...
7	10	Lorraine	Hannah	Silvers	17/08/1999	48 union street	Postrush	BT84 2TR	40027364	loraine47silvers...
8	11	Courtney	Lisa	Doherty	13/01/1994	11 jacobsLane	Ganagh	BT95 3RE	47326678	courtney20@gu...
9	12	Patrick	Samuel	Roberts	20/02/1973	92 Ballway	Ballybovey	BT05 5RF	30283736	patrick26@yahoo...
10	13	Matthew	Nathan	Jones	26/06/1975	45 tinkers street	Portstewart	BT82 7EQ	46335362	matthewjones...
11	14	Samantha	Alison	McFarlane	17/05/1982	231 Anderson A.	Ballymoney	BT00 3UN	45673623	sammym4@ya...
12	15	Linda	Hayley	Spears	04/04/1996	999 Goldhill	Ballymena	BT71 9YR	37434794	lindahspears29...
13	16	Lucy	Susan	Dysart	28/07/1991	52 Colour avenue	Coleraine	BT52 1QA	50382627	tonyhoward@g...
14	17	Anthony	Ryan	Howard	12/03/1990	32 Hillview	Postrush	BT95 3YU	50847464	tony.howard@g...
15	18	Cleo	Laura	Kane	23/02/2000	32 Hillview	Ballybovey	BT59 2EC	3882374	cleopatra6@hot...
16	19	Louise	Amanda	Doran	04/09/1994	39 Union street	Coleraine	BT51 3EH	754346934	loulodoran009...
17	20	Gerard	Josh	McDaid	25/04/1968	23 Chief Street	Bushmills	BT78 4YT	46637328	gerard500@gmail...
18	21	Kayla	Brooke	Black	13/10/1978	91 butchroad	Ballymoney	BT94 2YU	53752852	kaylablack@yahoo...
19	22	Kristeen	Casey	Dickenson	14/11/1979	19 Screenhill ro...	BallyCastle	BT39 5GF	53278523	Kdickenson@Y...
20	23	Leanna	Skylar	Philips	01/01/1992	124 Strand Park	Ganagh	BT43 9KV	96854746	LeaSky42@Hot...
21	24	Duke	Kassy	Ewart	25/12/1998	220 Anderson A.	Postrush	BT59 4RM	57325632	DukeEwart@G...
22	25	Alan	Shantelle	Blue	21/08/2995	60 Tullyerton D...	BallyBovey	BT99 0CJ	53275923	AlanBlue@Yah...
23	26	Reginald	Jessa	Stevenson	05/07/1982	14 Gelbe Park	Coleraine	BT46 2HD	1312334	rs24@gmail.co...
24	27	Frederica	Trevelyan	Cobb	09/03/2000	84 Crescent Road	Coleraine	BT59 5GL	70329356	FredCobb@Hot...
25	o	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

2.10[A][1] Screenshot B



The application window has a title bar 'frmScheduleTable'. The main area contains several input fields and dropdown menus:

- Student Details:** StudentID: 6, First Name: Daniel, Surname: Waters, Contact Number: 37719048.
- Teacher Details:** TeacherID: 1, First Name: Sandra, Surname: Smith, Specialisation: String, Room: 1.
- Grade Details:** GradeID: 4, Grade: Diploma.
- ScheduleID:** 11.

On the right side, there is a navigation section with buttons for 'Search for Student' (set to Adam Ackles), 'previous Student Record' and 'Next Student Record', and buttons for 'Add new Student', 'Update Lesson Record', 'Remove Lesson Record', 'Return to Calendar [Placeholder]', and 'Return to Main Menu'.

At the bottom, a table displays the schedule details:

ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date
11	6	1	3	30 Weeks	28/10/2016	Monday	09:30	26/05/2017

2.10[A][1] Screenshot C

Scheduled Lessons List

StudentID:	10																											
Student Details																												
First Name:	Lorraine																											
Surname:	Silvers																											
Contact Number:	40027364																											
Grade Details																												
GradeID:	1																											
Grade:	Beginner																											
ScheduleID:	8																											
<table border="1"> <thead> <tr> <th>ScheduleID</th> <th>StudentID</th> <th>TeacherID</th> <th>LessonsPurchasedID</th> <th>Number_Of_Weeks</th> <th>Start_Date</th> <th>Booked_Day(s)</th> <th>Booked_Time</th> <th>End_Date</th> </tr> </thead> <tbody> <tr> <td>8</td> <td>10</td> <td>4</td> <td>3</td> <td>30 Weeks</td> <td>27/10/2016</td> <td>Thursday</td> <td>09:30</td> <td>25/05/2017</td> </tr> <tr> <td>*</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>		ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date	8	10	4	3	30 Weeks	27/10/2016	Thursday	09:30	25/05/2017	*								
ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date																				
8	10	4	3	30 Weeks	27/10/2016	Thursday	09:30	25/05/2017																				
*																												

Teacher Details

TeacherID:	4
First Name:	Nathan
Surname:	Jones
Specialisation:	Percussion
Room:	4

Actions

- Search for Student: Adam Ackles
- previous Student Record
- Next Student Record
- Add new Student
- Update Lesson Record
- Remove Lesson Record
- Return to Calendar [Placeholder]
- Return to Main Menu

2.10[A][1] Screenshot D

Scheduled Lessons List

StudentID:	13																											
Student Details																												
First Name:	Matthew																											
Surname:	Jones																											
Contact Number:	46335362																											
Grade Details																												
GradeID:	3																											
Grade:	Advanced																											
ScheduleID:	25																											
<table border="1"> <thead> <tr> <th>ScheduleID</th> <th>StudentID</th> <th>TeacherID</th> <th>LessonsPurchasedID</th> <th>Number_Of_Weeks</th> <th>Start_Date</th> <th>Booked_Day(s)</th> <th>Booked_Time</th> <th>End_Date</th> </tr> </thead> <tbody> <tr> <td>25</td> <td>13</td> <td>2</td> <td>3</td> <td>30 Weeks</td> <td>01/11/2016</td> <td>Monday</td> <td>15:00</td> <td>30/05/2017</td> </tr> <tr> <td>*</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>		ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date	25	13	2	3	30 Weeks	01/11/2016	Monday	15:00	30/05/2017	*								
ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date																				
25	13	2	3	30 Weeks	01/11/2016	Monday	15:00	30/05/2017																				
*																												

Teacher Details

TeacherID:	2
First Name:	Carey
Surname:	Gamble
Specialisation:	Wood Wind
Room:	2

Actions

- Search for Student: Adam Ackles
- previous Student Record
- Next Student Record
- Add new Student
- Update Lesson Record
- Remove Lesson Record
- Return to Calendar [Placeholder]
- Return to Main Menu

2.10[A][2] – Grade Information**2.10[A][2] Screenshot A**

GradeID	GradeLevel	Grade Fee
1	Beginner	10
2	Intermediate	15
3	Advanced	20
4	Diploma	25
0	NULL	NULL

2.10[A][2] Screenshot B

ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date
11	6	1	3	30 Weeks	28/10/2016	Monday	09:30	26/05/2017

2.10[A][2] Screenshot C

Scheduled Lessons List

StudentID:	10																											
Student Details First Name: Lorraine Surname: Silvers Contact Number: 40027364																												
Teacher Details TeacherID: 4 First Name: Nathan Surname: Jones Specialisation: Percussion Room: 4																												
Grade Details GradeID: 1 Grade: Beginner																												
ScheduleID:	8																											
<table border="1"> <thead> <tr> <th>ScheduleID</th> <th>StudentID</th> <th>TeacherID</th> <th>LessonsPurchasedID</th> <th>Number_Of_Weeks</th> <th>Start_Date</th> <th>Booked_Day(s)</th> <th>Booked_Time</th> <th>End_Date</th> </tr> </thead> <tbody> <tr> <td>8</td> <td>10</td> <td>4</td> <td>3</td> <td>30 Weeks</td> <td>27/10/2016</td> <td>Thursday</td> <td>09:30</td> <td>25/05/2017</td> </tr> <tr> <td>*</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>		ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date	8	10	4	3	30 Weeks	27/10/2016	Thursday	09:30	25/05/2017	*								
ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date																				
8	10	4	3	30 Weeks	27/10/2016	Thursday	09:30	25/05/2017																				
*																												

Search for Student: Adam Ackles

← previous Student Record Next Student Record →

Add new Student
Update Lesson Record
Remove Lesson Record
Return to Calendar [Placeholder]
Return to Main Menu

2.10[A][2] Screenshot D

Scheduled Lessons List

StudentID:	13																											
Student Details First Name: Matthew Surname: Jones Contact Number: 46335362																												
Teacher Details TeacherID: 2 First Name: Carey Surname: Gamble Specialisation: Wood Wind Room: 2																												
Grade Details GradeID: 3 Grade: Advanced																												
ScheduleID:	25																											
<table border="1"> <thead> <tr> <th>ScheduleID</th> <th>StudentID</th> <th>TeacherID</th> <th>LessonsPurchasedID</th> <th>Number_Of_Weeks</th> <th>Start_Date</th> <th>Booked_Day(s)</th> <th>Booked_Time</th> <th>End_Date</th> </tr> </thead> <tbody> <tr> <td>25</td> <td>13</td> <td>2</td> <td>3</td> <td>30 Weeks</td> <td>01/11/2016</td> <td>Monday</td> <td>15:00</td> <td>30/05/2017</td> </tr> <tr> <td>*</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>		ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date	25	13	2	3	30 Weeks	01/11/2016	Monday	15:00	30/05/2017	*								
ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date																				
25	13	2	3	30 Weeks	01/11/2016	Monday	15:00	30/05/2017																				
*																												

2.10[A][3] – Lesson Bundle Information**2.10[A][3] Screenshot A**

	LessonBundleID	Lesson Bundle	Bundle Cost	Multiplier (Dis...)
▶	1	10 Lessons	... 10	1
	2	20 Lessons	... 20	0.95
	3	30 Lessons	... 30	0.9
◀	NULL	NULL	NULL	NULL

2.10[A][3] Screenshot B

Scheduled Lessons List

Student Details:

- Student ID: 6
- First Name: Daniel
- Surname: Waters
- Contact Number: 37719048

Teacher Details:

- TeacherID: 1
- First Name: Sandra
- Surname: Smith
- Specialisation: String
- Room: 1

Grade Details:

- GradeID: 4
- Grade: Diploma

ScheduleID: 11

ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date
11	6	1	3	30 Weeks	28/10/2016	Monday	09:30	26/05/2017

Actions:

- Search for Student: Adam Ackles
- previous Student Record
- Next Student Record
- Add new Student
- Update Lesson Record
- Remove Lesson Record
- Return to Calendar [Placeholder]
- Return to Main Menu

2.10[A][3] Screenshot C

Scheduled Lessons List

StudentID:	10																											
Student Details First Name: Lorraine Surname: Silvera Contact Number: 40027364																												
Teacher Details TeacherID: 4 First Name: Nathan Surname: Jones Specialisation: Percussion Room: 4																												
Grade Details GradeID: 1 Grade: Beginner																												
ScheduleID: 8																												
<table border="1"> <thead> <tr> <th>ScheduleID</th> <th>StudentID</th> <th>TeacherID</th> <th>LessonsPurchasedID</th> <th>Number_Of_Weeks</th> <th>Start_Date</th> <th>Booked_Day(s)</th> <th>Booked_Time</th> <th>End_Date</th> </tr> </thead> <tbody> <tr> <td>8</td> <td>10</td> <td>4</td> <td>3</td> <td>30 Weeks</td> <td>27/10/2016</td> <td>Thursday</td> <td>09:30</td> <td>25/05/2017</td> </tr> <tr> <td>*</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>		ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date	8	10	4	3	30 Weeks	27/10/2016	Thursday	09:30	25/05/2017	*								
ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date																				
8	10	4	3	30 Weeks	27/10/2016	Thursday	09:30	25/05/2017																				
*																												

2.10[A][3] Screenshot D

Scheduled Lessons List

StudentID:	13																											
Student Details First Name: Matthew Surname: Jones Contact Number: 46335362																												
Teacher Details TeacherID: 2 First Name: Carey Surname: Gamble Specialisation: Wood Wind Room: 2																												
Grade Details GradeID: 3 Grade: Advanced																												
ScheduleID: 25																												
<table border="1"> <thead> <tr> <th>ScheduleID</th> <th>StudentID</th> <th>TeacherID</th> <th>LessonsPurchasedID</th> <th>Number_Of_Weeks</th> <th>Start_Date</th> <th>Booked_Day(s)</th> <th>Booked_Time</th> <th>End_Date</th> </tr> </thead> <tbody> <tr> <td>25</td> <td>13</td> <td>2</td> <td>3</td> <td>30 Weeks</td> <td>01/11/2016</td> <td>Monday</td> <td>15:00</td> <td>30/05/2017</td> </tr> <tr> <td>*</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>		ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date	25	13	2	3	30 Weeks	01/11/2016	Monday	15:00	30/05/2017	*								
ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date																				
25	13	2	3	30 Weeks	01/11/2016	Monday	15:00	30/05/2017																				
*																												

2.10[A][4] – Purchased Lesson Information**2.10[A][4] Screenshot A**

SSMS Screenshot A: A screenshot of a Microsoft SQL Server Management Studio (SSMS) window showing a table named 'e_ScheduledLessons'. The table has columns: LessonPurchasedID, StudentID, LessonBundleID, Purchased_Date, Payment_Method, Payment_Recd, Payment_Recd_Dt, Total_Bundle_Amt, and Scheduled. The data shows multiple rows of lesson records. To the right of the table is a Windows Task Manager showing system performance metrics like CPU, Memory, and Disk usage.

2.10[A][4] Screenshot B

Windows Application Screenshot B: A screenshot of a Windows application window titled 'frmScheduleTable' showing a 'Scheduled Lessons List'. It includes sections for 'Student Details' (StudentID: 6, First Name: Daniel Waters, Contact Number: 37719048), 'Teacher Details' (TeacherID: 1, First Name: Sandra Smith, Specialisation: String, Room: 1), and 'Grade Details' (GradeID: 4, Grade: Diploma). On the right, there's a search bar ('Search for Student: Adam Ackles') and buttons for 'Add new Student', 'Update Lesson Record', 'Remove Lesson Record', 'Return to Calendar [Placeholder]', and 'Return to Main Menu'. Below the details is a table showing a single row of scheduled lesson information.

2.10[A][4] Screenshot C

Scheduled Lessons List

StudentID: 10	TeacherID: 4	Search for Student: Adam Ackles																											
Student Details		previous Student Record																											
First Name: Lorraine	First Name: Nathan	Next Student Record																											
Surname: Silvers	Surname: Jones																												
Contact Number: 40027364	Specialisation: Percussion	Add new Student																											
	Room: 4	Update Lesson Record																											
Grade Details		Remove Lesson Record																											
GradeID: 1	Grade: Beginner																												
ScheduleID: 8	Return to Calendar [Placeholder]																												
<table border="1"> <thead> <tr> <th>ScheduleID</th> <th>StudentID</th> <th>TeacherID</th> <th>LessonsPurchasedID</th> <th>Number_Of_Weeks</th> <th>Start_Date</th> <th>Booked_Day(s)</th> <th>Booked_Time</th> <th>End_Date</th> </tr> </thead> <tbody> <tr> <td>8</td> <td>10</td> <td>4</td> <td>3</td> <td>30 Weeks</td> <td>27/10/2016</td> <td>Thursday</td> <td>09:30</td> <td>25/05/2017</td> </tr> <tr> <td>*</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>			ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date	8	10	4	3	30 Weeks	27/10/2016	Thursday	09:30	25/05/2017	*								
ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date																					
8	10	4	3	30 Weeks	27/10/2016	Thursday	09:30	25/05/2017																					
*																													

2.10[A][4] Screenshot D

Scheduled Lessons List

StudentID: 13	TeacherID: 2	Search for Student: Adam Ackles																											
Student Details		previous Student Record																											
First Name: Matthew	First Name: Carey	Next Student Record																											
Surname: Jones	Surname: Gamble																												
Contact Number: 46335362	Specialisation: Wood Wind	Add new Student																											
	Room: 2	Update Lesson Record																											
Grade Details		Remove Lesson Record																											
GradeID: 3	Grade: Advanced																												
ScheduleID: 25	Return to Calendar [Placeholder]																												
<table border="1"> <thead> <tr> <th>ScheduleID</th> <th>StudentID</th> <th>TeacherID</th> <th>LessonsPurchasedID</th> <th>Number_Of_Weeks</th> <th>Start_Date</th> <th>Booked_Day(s)</th> <th>Booked_Time</th> <th>End_Date</th> </tr> </thead> <tbody> <tr> <td>25</td> <td>13</td> <td>2</td> <td>3</td> <td>30 Weeks</td> <td>01/11/2016</td> <td>Monday</td> <td>15:00</td> <td>30/05/2017</td> </tr> <tr> <td>*</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>			ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date	25	13	2	3	30 Weeks	01/11/2016	Monday	15:00	30/05/2017	*								
ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date																					
25	13	2	3	30 Weeks	01/11/2016	Monday	15:00	30/05/2017																					
*																													

*Process [B]) Record Navigation***2.10[B][1] – Student Search****2.10[B][1] Screenshot A**

Scheduled Lessons List

Student ID:	13																											
Student Details First Name: Matthew Surname: Jones Contact Number: 46335362																												
Teacher Details TeacherID: 2 First Name: Carey Surname: Gamble Specialisation: Wood Wind Room: 2																												
Grade Details GradeID: 3 Grade: Advanced																												
ScheduleID:	25																											
<table border="1"> <thead> <tr> <th>ScheduleID</th> <th>StudentID</th> <th>TeacherID</th> <th>LessonsPurchasedID</th> <th>Number_of_Weeks</th> <th>Start_Date</th> <th>Booked_Day(s)</th> <th>Booked_Time</th> <th>End_Date</th> </tr> </thead> <tbody> <tr> <td>25</td> <td>13</td> <td>2</td> <td>3</td> <td>30 Weeks</td> <td>01/11/2016</td> <td>Monday</td> <td>15:00</td> <td>30/05/2017</td> </tr> <tr> <td>*</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>		ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date	25	13	2	3	30 Weeks	01/11/2016	Monday	15:00	30/05/2017	*								
ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date																				
25	13	2	3	30 Weeks	01/11/2016	Monday	15:00	30/05/2017																				
*																												

Search for Student: Adam Ackles

Add new Record

Update Lesson Record

Remove Lesson Record

Return to Calendar

Return to Main Menu

2.10[B][1] Screenshot B

Scheduled Lessons List

Student ID:	4																		
Student Details First Name: Linda Surname: Simmons Contact Number: 70359371																			
Teacher Details TeacherID: <input type="text"/> First Name: <input type="text"/> Surname: <input type="text"/> Specialisation: <input type="text"/> Room: <input type="text"/>																			
Grade Details GradeID: 2 Grade: Intermediate																			
ScheduleID:	<input type="text"/>																		
<table border="1"> <thead> <tr> <th>ScheduleID</th> <th>StudentID</th> <th>TeacherID</th> <th>LessonsPurchasedID</th> <th>Number_of_Weeks</th> <th>Start_Date</th> <th>Booked_Day(s)</th> <th>Booked_Time</th> <th>End_Date</th> </tr> </thead> <tbody> <tr> <td>*</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>		ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date	*								
ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date											
*																			

Search for Student: Linda Simmons

Next Student Record

Add new Student Record

Update Lesson Record

Remove Lesson Record

Return to Calendar [Placeholder]

Return to Main Menu

2.10[B][2] – Next Student**2.10[B][2] Screenshot A**

Scheduled Lessons List

StudentID:	4																		
Student Details																			
First Name:	Linda																		
Surname:	Simmons																		
Contact Number:	70359371																		
Grade Details																			
GradeID:	2																		
Grade:	Intermediate																		
ScheduleID:																			
<table border="1"> <thead> <tr> <th>ScheduleID</th> <th>StudentID</th> <th>TeacherID</th> <th>LessonsPurchasedID</th> <th>Number_of_Weeks</th> <th>Start_Date</th> <th>Booked_Day(s)</th> <th>Booked_Time</th> <th>End_Date</th> </tr> </thead> <tbody> <tr> <td>*</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>		ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date	*								
ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date											
*																			

Teacher Details

TeacherID:	
First Name:	
Surname:	
Specialisation:	
Room:	

Actions

- Search for Student: Linda Simmons
- previous Student Record
- Next Student Record
- Add new Student
- Update Lesson Record
- Remove Lesson Record
- Return to Calendar [Placeholder]
- Return to Main Menu

2.10[B][2] Screenshot B

Scheduled Lessons List

StudentID:	5																		
Student Details																			
First Name:	Trevor																		
Surname:	Simmons																		
Contact Number:	74920033																		
Grade Details																			
GradeID:	1																		
Grade:	Beginner																		
ScheduleID:																			
<table border="1"> <thead> <tr> <th>ScheduleID</th> <th>StudentID</th> <th>TeacherID</th> <th>LessonsPurchasedID</th> <th>Number_of_Weeks</th> <th>Start_Date</th> <th>Booked_Day(s)</th> <th>Booked_Time</th> <th>End_Date</th> </tr> </thead> <tbody> <tr> <td>*</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>		ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date	*								
ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date											
*																			

Teacher Details

TeacherID:	
First Name:	
Surname:	
Specialisation:	
Room:	

Actions

- Search for Student: Linda Simmons
- previous Student Record
- Next Student Record
- Add new Student
- Update Lesson Record
- Remove Lesson Record
- Return to Calendar [Placeholder]
- Return to Main Menu

2.10[B][3] – Previous Student**2.10[B][3] Screenshot A**

Scheduled Lessons List

Student ID:	5
Student Details First Name: Trevor Surname: Simmons Contact Number: 74920033	
Teacher Details TeacherID: <input type="text"/> First Name: <input type="text"/> Surname: <input type="text"/> Specialisation: <input type="text"/> Room: <input type="text"/>	
Grade Details GradeID: 1 Grade: Beginner	
ScheduleID: <input type="text"/>	
ScheduleID StudentID TeacherID LessonsPurchasedID Number_Of_Weeks Start_Date Booked_Day(s) Booked_Time End_Date * <input type="text"/>	

Search for Student: Linda Simmons
 previous Student Record Next Student Record
 Add new Student Update Lesson Record
 Remove Lesson Record
 Return to Calendar [Placeholder]
 Return to Main Menu

2.10[B][3] Screenshot B

Scheduled Lessons List

Student ID:	4
Student Details First Name: Linda Surname: Simmons Contact Number: 70359371	
Teacher Details TeacherID: <input type="text"/> First Name: <input type="text"/> Surname: <input type="text"/> Specialisation: <input type="text"/> Room: <input type="text"/>	
Grade Details GradeID: 2 Grade: Intermediate	
ScheduleID: <input type="text"/>	
ScheduleID StudentID TeacherID LessonsPurchasedID Number_Of_Weeks Start_Date Booked_Day(s) Booked_Time End_Date * <input type="text"/>	

Search for Student: Linda Simmons
 previous Student Record Next Student Record
 Add new Student Update Lesson Record
 Remove Lesson Record
 Return to Calendar [Placeholder]
 Return to Main Menu

2.10[B][4] – First Student**2.10[B][4] Screenshot A**

Purchased Lessons

Student ID:	27																		
Student Details																			
First Name:	Frederica																		
Surname:	Cobb																		
Contact Number:	70329356																		
Grade Details																			
GradeID:	3																		
Grade:	Advanced																		
Purchase ID:																			
<table border="1"> <thead> <tr> <th>LessonsPurchasedID</th> <th>StudentID</th> <th>LessonBundleID</th> <th>Purchased_Date</th> <th>Payment_Method</th> <th>Payment_Received</th> <th>Payment_Received Date</th> <th>Total_Bundle_Cost</th> <th>Scheduled</th> </tr> </thead> <tbody> <tr> <td>*</td> <td></td> <td></td> <td></td> <td></td> <td><input type="checkbox"/></td> <td></td> <td></td> <td><input type="checkbox"/></td> </tr> </tbody> </table>		LessonsPurchasedID	StudentID	LessonBundleID	Purchased_Date	Payment_Method	Payment_Received	Payment_Received Date	Total_Bundle_Cost	Scheduled	*					<input type="checkbox"/>			<input type="checkbox"/>
LessonsPurchasedID	StudentID	LessonBundleID	Purchased_Date	Payment_Method	Payment_Received	Payment_Received Date	Total_Bundle_Cost	Scheduled											
*					<input type="checkbox"/>			<input type="checkbox"/>											

Lesson Bundle Details

BundleID:	1
Lesson Bundle:	10 Lessons
Bundle Cost:	10
Cost Multiplier:	1.00

Search for Student: Adam Ackles

Action Buttons:

- Add New Purchase
- Update Student Record
- Remove Lesson Record

Return to Main Menu

2.10[B][4] Screenshot B

Purchased Lessons

Student ID:	4																																				
Student Details																																					
First Name:	Linda																																				
Surname:	Simmons																																				
Contact Number:	70359371																																				
Grade Details																																					
GradeID:	2																																				
Grade:	Intermediate																																				
Purchase ID:																																					
<table border="1"> <thead> <tr> <th>LessonsPurchasedID</th> <th>StudentID</th> <th>LessonBundleID</th> <th>Purchased_Date</th> <th>Payment_Method</th> <th>Payment_Received</th> <th>Payment_Received Date</th> <th>Total_Bundle_Cost</th> <th>Scheduled</th> </tr> </thead> <tbody> <tr> <td>13</td> <td>4</td> <td>2</td> <td>12/08/2016</td> <td>Cash</td> <td><input checked="" type="checkbox"/></td> <td>12/08/2016</td> <td></td> <td><input type="checkbox"/></td> </tr> <tr> <td>31</td> <td>4</td> <td>3</td> <td>01/08/2016</td> <td>Cash</td> <td><input checked="" type="checkbox"/></td> <td>01/09/2016</td> <td></td> <td><input type="checkbox"/></td> </tr> <tr> <td>*</td> <td></td> <td></td> <td></td> <td></td> <td><input type="checkbox"/></td> <td></td> <td></td> <td><input type="checkbox"/></td> </tr> </tbody> </table>		LessonsPurchasedID	StudentID	LessonBundleID	Purchased_Date	Payment_Method	Payment_Received	Payment_Received Date	Total_Bundle_Cost	Scheduled	13	4	2	12/08/2016	Cash	<input checked="" type="checkbox"/>	12/08/2016		<input type="checkbox"/>	31	4	3	01/08/2016	Cash	<input checked="" type="checkbox"/>	01/09/2016		<input type="checkbox"/>	*					<input type="checkbox"/>			<input type="checkbox"/>
LessonsPurchasedID	StudentID	LessonBundleID	Purchased_Date	Payment_Method	Payment_Received	Payment_Received Date	Total_Bundle_Cost	Scheduled																													
13	4	2	12/08/2016	Cash	<input checked="" type="checkbox"/>	12/08/2016		<input type="checkbox"/>																													
31	4	3	01/08/2016	Cash	<input checked="" type="checkbox"/>	01/09/2016		<input type="checkbox"/>																													
*					<input type="checkbox"/>			<input type="checkbox"/>																													

Lesson Bundle Details

BundleID:	2
Lesson Bundle:	20 Lessons
Bundle Cost:	20
Cost Multiplier:	0.95

Search for Student: Adam Ackles

Action Buttons:

- Add New Purchase
- Update Student Record
- Remove Lesson Record

Return to Main Menu

2.10[B][5] – Last Student**2.10[B][5] Screenshot A**

Purchased Lessons

Student ID:	4																																				
Student Details First Name: Linda Surname: Simmons Contact Number: 70359371																																					
Lesson Bundle Details BundleID: 2 Lesson Bundle: 20 Lessons Bundle Cost: 20 Cost Multiplier: 0.95																																					
Grade Details GradeID: 2 Grade: Intermediate																																					
Purchase ID: 13																																					
<table border="1"> <thead> <tr> <th>LessonsPurchasedID</th> <th>StudentID</th> <th>LessonBundleID</th> <th>Purchased_Date</th> <th>Payment_Method</th> <th>Payment_Received</th> <th>Payment_Received Date</th> <th>Total_Bundle_Cost</th> <th>Scheduled</th> </tr> </thead> <tbody> <tr> <td>13</td> <td>4</td> <td>2</td> <td>12/08/2016</td> <td>Cash</td> <td><input checked="" type="checkbox"/></td> <td>12/08/2016</td> <td></td> <td><input type="checkbox"/></td> </tr> <tr> <td>31</td> <td>4</td> <td>3</td> <td>01/08/2016</td> <td>Cash</td> <td><input checked="" type="checkbox"/></td> <td>01/09/2016</td> <td></td> <td><input type="checkbox"/></td> </tr> <tr> <td>*</td> <td></td> <td></td> <td></td> <td></td> <td><input type="checkbox"/></td> <td></td> <td></td> <td><input type="checkbox"/></td> </tr> </tbody> </table>		LessonsPurchasedID	StudentID	LessonBundleID	Purchased_Date	Payment_Method	Payment_Received	Payment_Received Date	Total_Bundle_Cost	Scheduled	13	4	2	12/08/2016	Cash	<input checked="" type="checkbox"/>	12/08/2016		<input type="checkbox"/>	31	4	3	01/08/2016	Cash	<input checked="" type="checkbox"/>	01/09/2016		<input type="checkbox"/>	*					<input type="checkbox"/>			<input type="checkbox"/>
LessonsPurchasedID	StudentID	LessonBundleID	Purchased_Date	Payment_Method	Payment_Received	Payment_Received Date	Total_Bundle_Cost	Scheduled																													
13	4	2	12/08/2016	Cash	<input checked="" type="checkbox"/>	12/08/2016		<input type="checkbox"/>																													
31	4	3	01/08/2016	Cash	<input checked="" type="checkbox"/>	01/09/2016		<input type="checkbox"/>																													
*					<input type="checkbox"/>			<input type="checkbox"/>																													

Search for Student: Adam Ackles
 previous Student Record Next Student Record
 Add New Purchase Update Student Record Remove Lesson Record
 Return to Main Menu

2.10[B][5] Screenshot B

Purchased Lessons

Student ID:	27																		
Student Details First Name: Frederica Surname: Cobb Contact Number: 70329356																			
Lesson Bundle Details BundleID: <input type="text"/> Lesson Bundle: <input type="text"/> Bundle Cost: <input type="text"/> Cost Multiplier: <input type="text"/>																			
Grade Details GradeID: 3 Grade: Advanced																			
Purchase ID: <input type="text"/>																			
<table border="1"> <thead> <tr> <th>LessonsPurchasedID</th> <th>StudentID</th> <th>LessonBundleID</th> <th>Purchased_Date</th> <th>Payment_Method</th> <th>Payment_Received</th> <th>Payment_Received Date</th> <th>Total_Bundle_Cost</th> <th>Scheduled</th> </tr> </thead> <tbody> <tr> <td>*</td> <td></td> <td></td> <td></td> <td></td> <td><input type="checkbox"/></td> <td></td> <td></td> <td><input type="checkbox"/></td> </tr> </tbody> </table>		LessonsPurchasedID	StudentID	LessonBundleID	Purchased_Date	Payment_Method	Payment_Received	Payment_Received Date	Total_Bundle_Cost	Scheduled	*					<input type="checkbox"/>			<input type="checkbox"/>
LessonsPurchasedID	StudentID	LessonBundleID	Purchased_Date	Payment_Method	Payment_Received	Payment_Received Date	Total_Bundle_Cost	Scheduled											
*					<input type="checkbox"/>			<input type="checkbox"/>											

Search for Student: Adam Ackles
 previous Student Record Next Student Record
 Add New Purchase Update Student Record Remove Lesson Record
 Return to Main Menu

*Process [C]) Form Navigation***2.10[C][1] – Add New Purchased Lesson Record****2.10[C][1] Screenshot A**

Purchased Lessons

Student Details:
First Name: Linda
Surname: Simmons
Contact Number: 70359371

Lesson Bundle Details:
BundleID: 3
Lesson Bundle: 30 Lessons
Bundle Cost: 30
Cost Multiplier: 0.9

Grade Details:
GradeID: 2
Grade: Intermediate

Purchase ID: 31

LessonsPurchasedID	StudentID	LessonBundleID	Purchased_Date	Payment_Method	Payment_Received	Payment_Received Date	Total_Bundle_Cost	Scheduled
31	4	3	01/08/2016	Cash	<input checked="" type="checkbox"/>	01/09/2016	405	<input type="checkbox"/>

Buttons:
Search for Student: Adam Ackles
previous Student Record
Next Student Record
Add New Purchase
Update Student Record
Remove Lesson Record
Return to Main Menu

2.10[C][1] Screenshot B

New Lesson Purchased:

Purchase New Lesson **Return to Purchased Lessons Table**

Fields:
Student ID:
Purchase ID:
Purchase Date:
Payment Method:
Payment Received:
payment Receipt Date:

2.10[C][2] – Update Purchased Lesson Record (Record Selected)**2.10[C][2] Screenshot A**

The screenshot shows the 'Purchased Lessons' window. At the top left is a logo for 'The Musical School Of Music'. The main title 'Purchased Lessons' is centered at the top. On the left, there are two sets of input fields: 'Student Details' (First Name: Linda, Surname: Simmons, Contact Number: 70359371) and 'Lesson Bundle Details' (Bundle ID: 3, Lesson Bundle: 30 Lessons, Bundle Cost: 30, Cost Multiplier: 0.9). Below these are 'Grade Details' (Grade ID: 2, Grade: Intermediate). In the center, a table displays a single record: Purchase ID 31, Student ID 4, Lesson Bundle ID 3, Purchased Date 01/08/2016, Payment Method Cash, Payment Received checked, Payment Received Date 01/09/2016, Total_Bundle_Cost 405, and Scheduled unchecked. To the right is a sidebar with buttons for 'Add New Purchase', 'Update Student Record', 'Remove Lesson Record', and 'Return to Main Menu'. A search bar at the top right shows 'Search for Student: Adam Ackles'.

2.10[C][2] Screenshot B

The screenshot shows the 'New Lesson Purchased:' window. The title is at the top. Below it are two buttons: 'Purchase New Lesson' on the left and 'Return to Purchased Lessons Table' on the right. The main area contains several input fields: Purchase ID (31), Student ID (4), Purchase ID (3), Purchase Date (01/08/2016), Payment Method (Cash), Payment Received (True), and payment Receipt Date (01/09/2016).

2.10[C][3] – Update Purchased Lesson Record (No Record Selected)**2.10[C][3] Screenshot A**

The screenshot shows a software application window titled "PurchasedLessonsBundles". The main title bar says "Purchased Lessons". The interface includes several input fields and buttons:

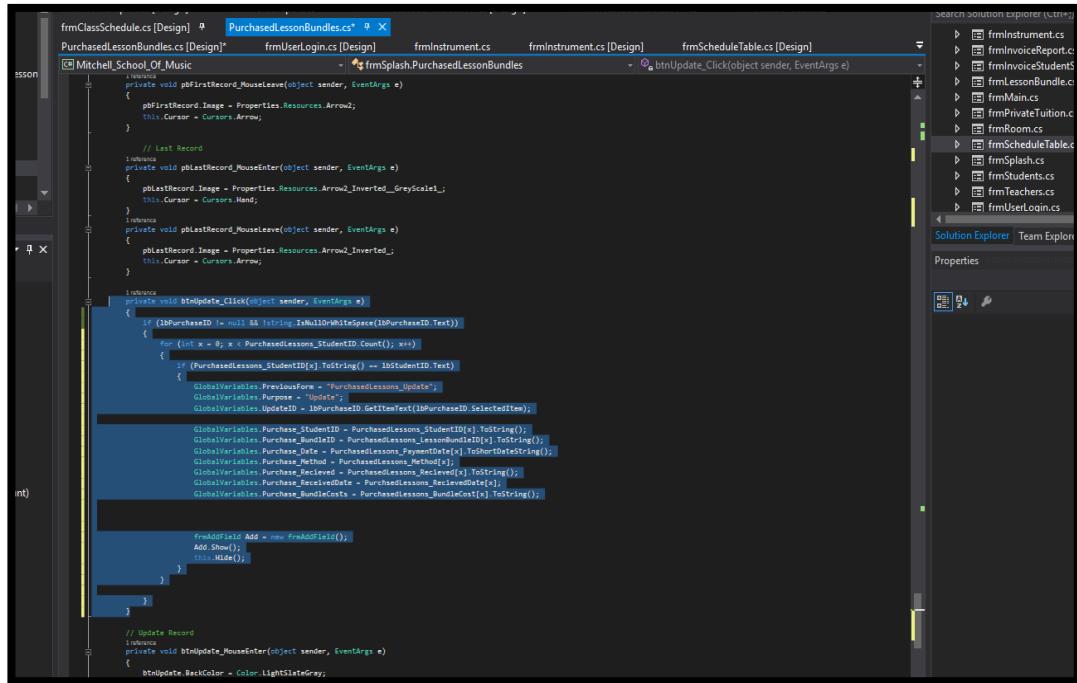
- Student Details:** Student ID: 11, First Name: Courtney, Surname: Doherty, Contact Number: 47326678.
- Lesson Bundle Details:** BundleID: [empty], Lesson Bundle: [empty], Bundle Cost: [empty], Cost Multiplier: [empty].
- Grade Details:** GradeID: 1, Grade: Beginner.
- Purchase ID:** [empty].
- Buttons:** Search for Students (with dropdown menu), previous Student Record, Next Student Record, Add New Purchase, Update Student Record, Remove Lesson Record, and Return to Main Menu.
- Data Grid:** A table showing a single row of data:

LessonsPurchasedID	StudentID	LessonBundleID	Purchased_Date	Payment_Method	Payment_Received	Payment_Received Date	Total_Bundle_Cost	Scheduled
*					<input type="checkbox"/>			<input type="checkbox"/>

2.10[C][3] Screenshot B

The screenshot shows a software application window with a dark blue header bar. The header bar features a logo of a guitar and the text "[Placeholder]".

The main content area contains the text "[Placeholder]" and a red button labeled "Return to [Placeholder]".

2.10[C][3] Screenshot C


```

private void pbFirstRecord_MouseLeave(object sender, EventArgs e)
{
    pbFirstRecord.Image = Properties.Resources.Arrow2;
    this.Cursor = Cursors.Arrow;
}

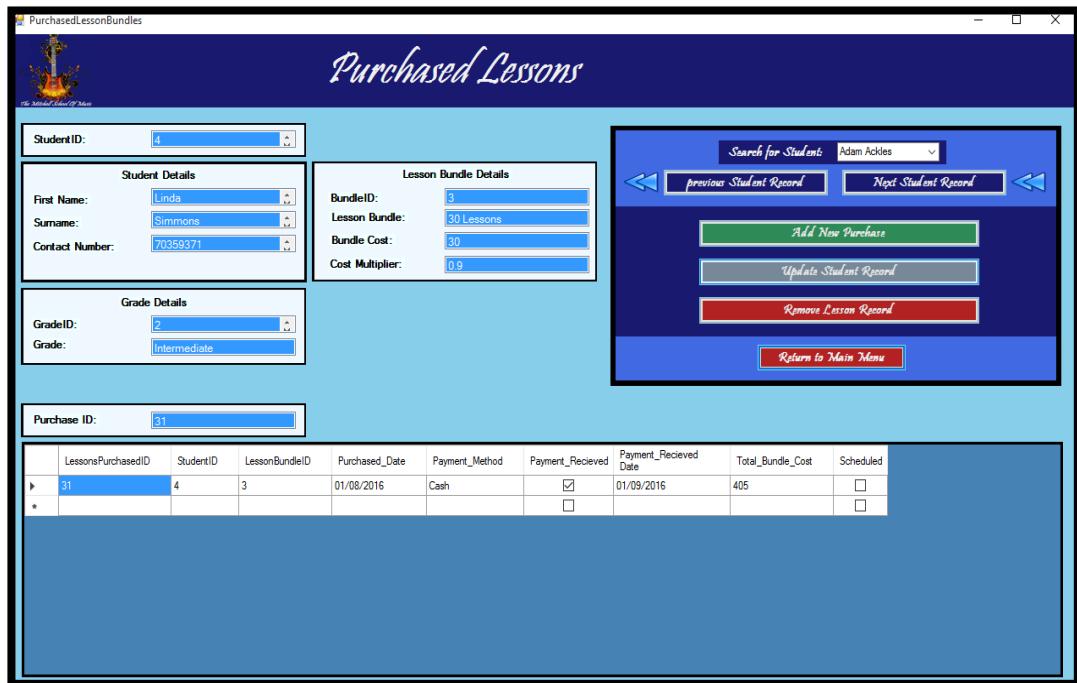
// Last Record
private void pbLastRecord_MouseEnter(object sender, EventArgs e)
{
    pbLastRecord.Image = Properties.Resources.Arrow2_Inverted_GreyScale1;
    this.Cursor = Cursors.Hand;
}

private void pbLastRecord_MouseLeave(object sender, EventArgs e)
{
    pbLastRecord.Image = Properties.Resources.Arrow2_Inverted;
    this.Cursor = Cursors.Arrow;
}

private void btUpdate_Click(object sender, EventArgs e)
{
    if (!tbPurchaseID.Text.IsNullOrWhiteSpace(tbPurchaseID.Text))
    {
        for (int x = 0; x < PurchasedLessons.StudentID.Count(); x++)
        {
            if (PurchasedLessons.StudentID[x].ToString() == tbStudentID.Text)
            {
                if (PurchasedLessons.StudentID[x].ToString() == tbPurchaseID.Text)
                {
                    GlobalVariables.PreviousForm = "PurchasedLessons_Update";
                    GlobalVariables.Purpose = "Update";
                    GlobalVariables.UpdateID = tbPurchaseID.SelectedItem;
                    GlobalVariables.Purchase_StudentID = PurchasedLessons.StudentID[x].ToString();
                    GlobalVariables.Purchase_BundleID = PurchasedLessons.LessonBundleID[x].ToString();
                    GlobalVariables.Purchase_Date = PurchasedLessons.PaymentDate[x].ToShortDateString();
                    GlobalVariables.Purchase_Method = PurchasedLessons.Method[x];
                    GlobalVariables.Purchase_Amount = PurchasedLessons.Amount[x].ToString();
                    GlobalVariables.Purchase_ReceivedDate = PurchasedLessons.ReceivedDate[x];
                    GlobalVariables.Purchase_BundleCosts = PurchasedLessons.BundleCost[x].ToString();
                }
            }
        }
    }
}

private void btUpdate_MouseEnter(object sender, EventArgs e)
{
    btUpdate.BackColor = Color.LightSlateGray;
}

```

2.10[C][3] Screenshot D

2.10[C][3] Screenshot E

New Lesson Purchased:

Purchase ID: 31

Student ID: 4

Purchase ID: 3

Purchase Date: 01/08/2016

Payment Method: Cash

Payment Received: True

payment Receipt Date: 09/2016

[Purchase New Lesson](#) [Return to Purchased Lessons Table](#)

**2.10[C][4] – Delete Purchased Lesson Record [User Logged In]
(No Purchased Lesson Selected)****2.10[C][4] Screenshot A**

Purchased Lessons

Student ID: 27

Student Details

First Name: Frederica

Surname: Cobb

Contact Number: 70329356

Grade Details

GradeID: 3

Grade: Advanced

Lesson Bundle Details

BundleID: [empty]

Lesson Bundle: [empty]

Bundle Cost: [empty]

Cost Multiplier: [empty]

Search for Student: Adam Ackles

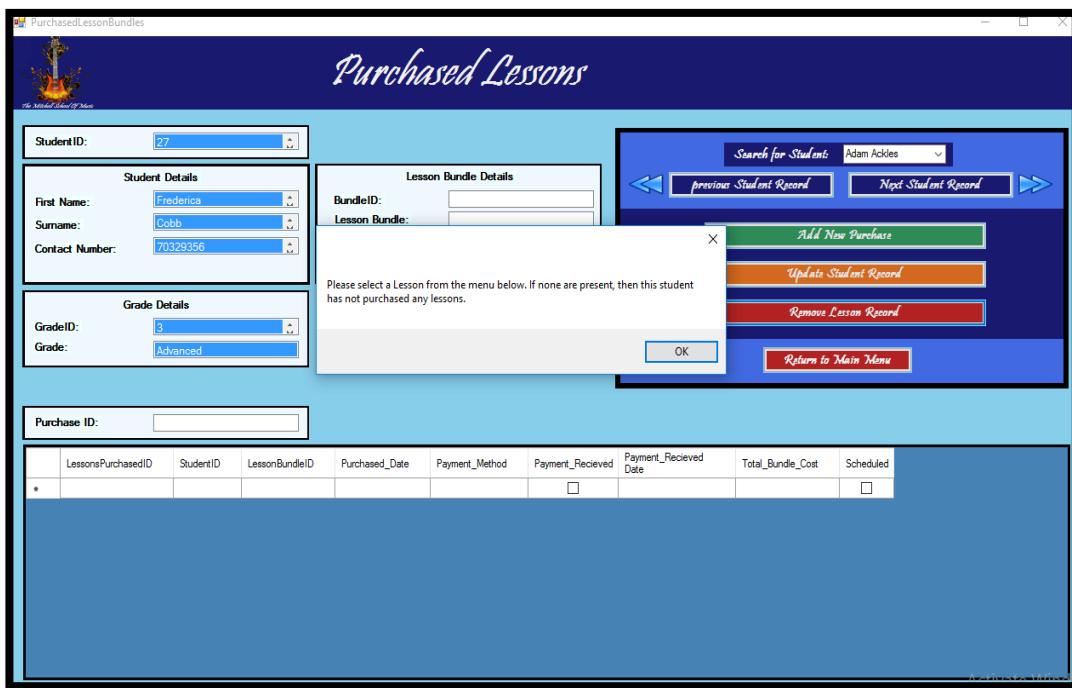
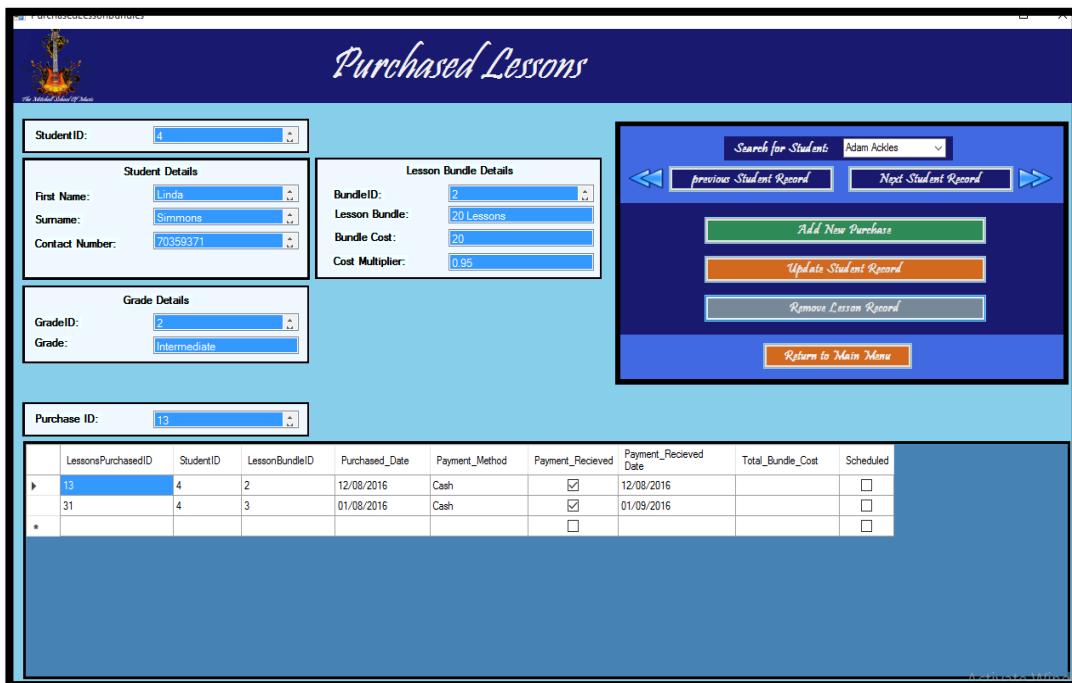
Add New Purchase

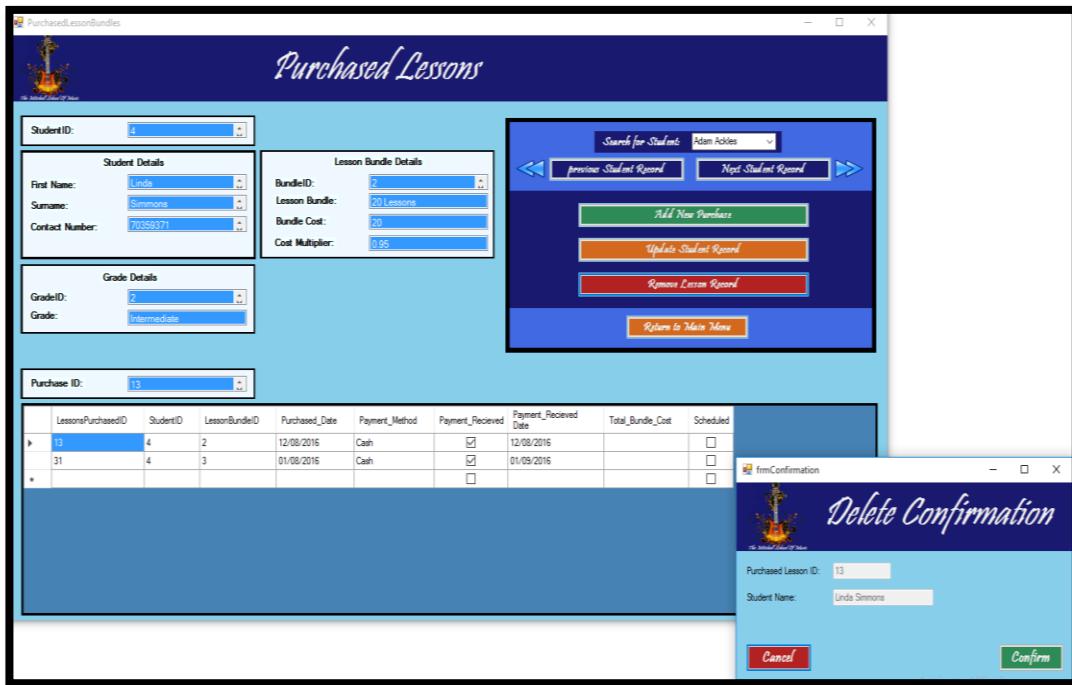
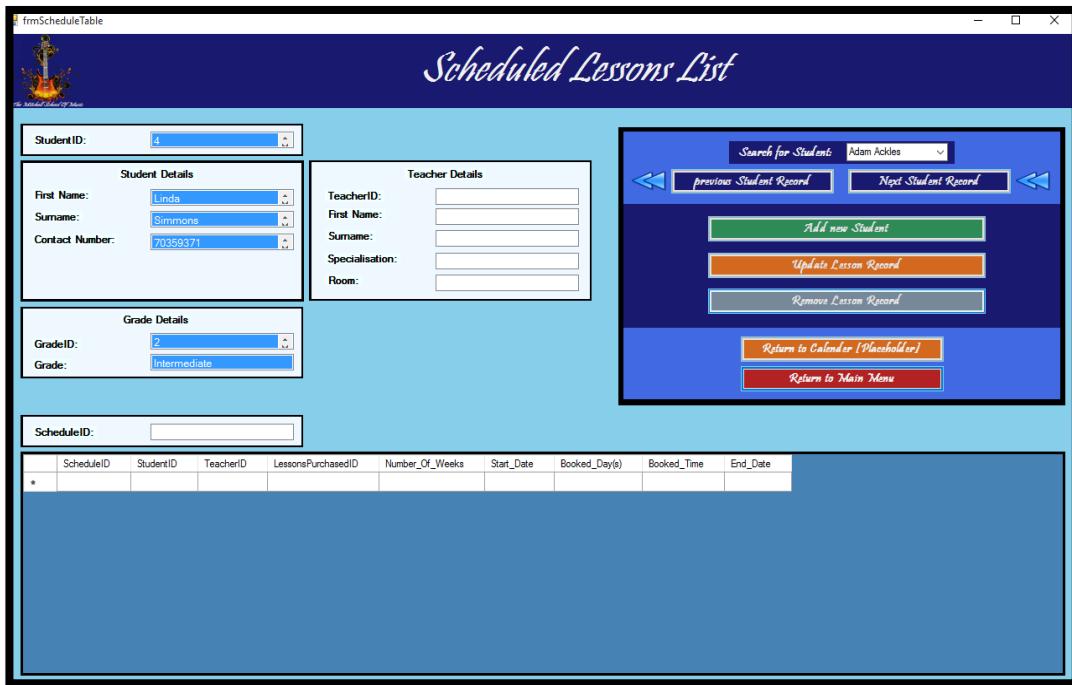
Update Student Record

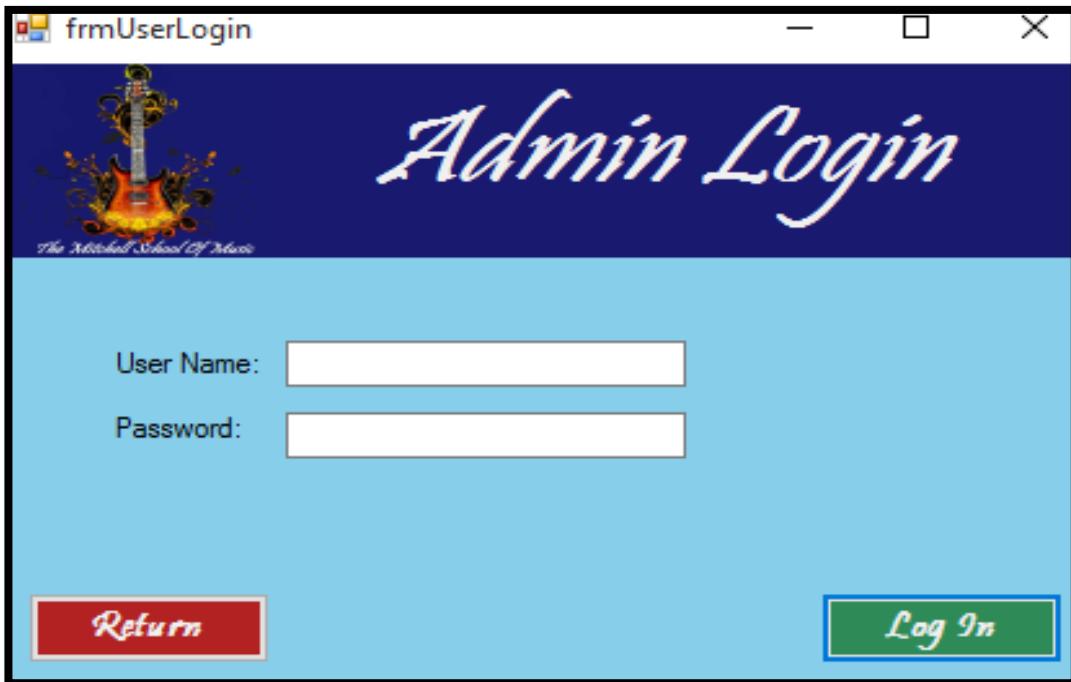
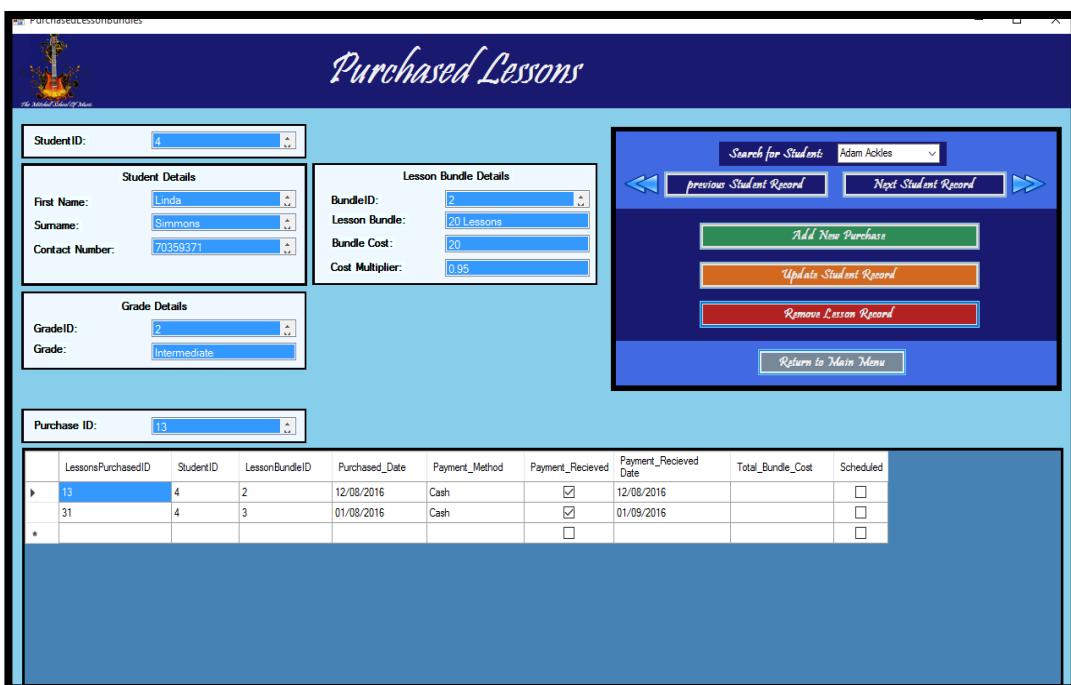
Remove Lesson Record

Return to Main Menu

LessonsPurchasedID	StudentID	LessonBundleID	Purchased_Date	Payment_Method	Payment_Received	Payment_Received_Date	Total_Bundle_Cost	Scheduled
*					<input type="checkbox"/>			<input type="checkbox"/>

2.10[C][4] Screenshot B**2.10[C][5] – Delete Purchased Lesson Record [User Logged In]
(Purchased Lesson Selected)****2.10[C][5] Screenshot A**

2.10[C][5] Screenshot B**2.10[C][6] – Delete Purchased Lesson Record [No User Logged In]****2.10[C][6] Screenshot A**

2.10[C][6] Screenshot B**2.10[C][7] – Return To Main Menu****2.10[C][7] Screenshot A**

2.10[C][7] Screenshot B**2.11) Scheduled Lessons Form***Process [A]) Information Display***2.11[A][1] – Student Information****2.11[A][1] Screenshot A**

PurchasedLessonBundles.cs [Design]												
StudentID	First_Name	Other_Name(s)	Surname	DateOfBirth	Address	Town	PostCode	Contact_Num...	Email_Address	GradelD	InstrumentID	Tuition_Fee_R...
1	Linda	Louise	Simmons	20/02/1998	14 Greenroad	Colecrane	BT51 2HE	70359371	Linda54@gmail...	2	1	True
5	Trevor	Luke	Simmons	20/08/1997	22 The Glebe	Ballymoney	BT32 3ND	74920033	Trevorimmons...	1	4	True
6	Daniel	Robert	Waters	01/02/1970	199 Pacroad	Ballymoney	BT47 3MF	37719048	danielwaters99...	4	3	True
7	Adam	Brent	Ackles	30/06/1993	731 bottlane	Garragh	BT43 2HT	46299567	adambrent@ya...	2	2	False
8	Jenson	James	Tweed	22/12/1990	71 glebe avenue	Ballycastle	BT37 5GW	33923231	jrtweed@hotmail...	4	5	True
9	Diana	Ann	James	14/04/1985	6 Polyroad	Bushmills	BT83 5FE	40039237	dianajames@ya...	4	6	True
10	Lorraine	Hannah	Silvers	17/08/1999	48 union street	Postrush	BT84 2TR	40027364	loraine47silvers...	1	7	False
11	Courtney	Lisa	Doherty	13/01/1994	11 jacobsLane	Garragh	BT95 3RE	47326678	courtney2009@...	1	8	True
12	Patrick	Samuel	Roberts	20/02/1973	92 Ballyway	Ballybogey	BT05 8RF	30283736	patrick26@yahoo...	2	9	True
13	Matthew	Nathan	Jones	26/06/1975	45 tinkies street	Portstewart	BT82 7EQ	46335362	matthewjones...	3	2	False
14	Samantha	Alison	Mc farlane	17/05/1982	231 Anderson A...	Ballymoney	BT00 3UN	45673632	sammym4@yahoo...	2	4	True
15	Linda	Hayley	Spears	04/04/1996	999 Goldhill	Ballymena	BT71 9YR	37434794	lindahspears39...	3	5	False
16	Lucy	Susan	Dysart	28/07/1991	52 Colour avenue	Colecrane	BT52 1QA	50382627	tonyhoward@q...	4	6	True
17	Anthony	Ryan	Howard	12/03/1990	32 Hillview	Postrush	BT95 3VU	50947464	tonyhoward@g...	3	10	True
18	Cleo	Laura	Kane	23/02/2000	32 Hillview	Ballybogey	BT59 2EC	3892374	cleopatra66@hot...	4	2	True
19	Louise	Amanda	Doran	04/09/1994	39 Union street	Colecrane	BT51 3EH	75434934	louloudoran009...	3	5	True
20	Gerard	Josh	Mc daid	25/04/1968	23 Chief Street	Bushmills	BT78 4YT	46667328	gerard300@gm...	1	7	False
21	Kayla	Brooke	Black	13/10/1978	91 butchord	Ballymoney	BT94 2VU	53752852	kaylablack@ya...	2	6	False
22	Kristeen	Casey	Dickenson	14/11/1979	19 Screenhill ro...	BallyCastle	BT39 5GF	53278523	Kdickenson@Y...	2	10	False
23	Leanna	Skylar	Philips	01/01/1992	124 Strand Park	Garragh	BT43 9KV	96854746	LeaSky42@Hot...	1	3	False
24	Duke	Kassy	Ewart	25/12/1998	220 Anderson A...	Postrush	BT59 4RM	57324632	DukeEwart@G...	1	3	False
25	Alan	Shantelle	Blue	21/08/2995	60 Tullyarton D...	BallyBogey	BT99 0CJ	53275823	AlanBlue@Yah...	4	2	True
26	Reginald	Jessa	Stevenson	05/07/1982	14 Gelbe Park	Colecrane	BT46 2HD	13123554	rp24@gmail.co...	1	9	False
27	Frederica	Trevelyn	Cobb	09/03/2000	84 Crescent Road	Colecrane	BT59 5GL	70329356	FredCobb@Hot...	3	4	True
8	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

2.11[A][1] Screenshot B

Scheduled Lessons List

StudentID: 6	TeacherID: 1	Search for Student: Adam Ackles																											
Student Details		previous Student Record																											
First Name: Daniel	Surname: Waters	Next Student Record																											
Contact Number: 37719048																													
Teacher Details		Add new Student																											
GradeID: 4	Surname: Smith	Update Lesson Record																											
Grade: Diploma	Specialisation: String	Remove Lesson Record																											
ScheduleID: 11	Return to Calendar [Placeholder]																												
Return to Main Menu																													
<table border="1"> <thead> <tr> <th>ScheduleID</th> <th>StudentID</th> <th>TeacherID</th> <th>LessonsPurchasedID</th> <th>Number_of_Weeks</th> <th>Start_Date</th> <th>Booked_Day(s)</th> <th>Booked_Time</th> <th>End_Date</th> </tr> </thead> <tbody> <tr> <td>11</td> <td>6</td> <td>1</td> <td>3</td> <td>30 Weeks</td> <td>28/10/2016</td> <td>Monday</td> <td>09:30</td> <td>26/05/2017</td> </tr> <tr> <td>*</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>			ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date	11	6	1	3	30 Weeks	28/10/2016	Monday	09:30	26/05/2017	*								
ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date																					
11	6	1	3	30 Weeks	28/10/2016	Monday	09:30	26/05/2017																					
*																													

2.11[A][1] Screenshot C

Scheduled Lessons List

StudentID: 10	TeacherID: 4	Search for Student: Adam Ackles																											
Student Details		previous Student Record																											
First Name: Lorraine	Surname: Silvers	Next Student Record																											
Contact Number: 40027364																													
Teacher Details		Add new Student																											
GradeID: 1	Surname: Jones	Update Lesson Record																											
Grade: Beginner	Specialisation: Percussion	Remove Lesson Record																											
ScheduleID: 8	Return to Calendar [Placeholder]																												
Return to Main Menu																													
<table border="1"> <thead> <tr> <th>ScheduleID</th> <th>StudentID</th> <th>TeacherID</th> <th>LessonsPurchasedID</th> <th>Number_of_Weeks</th> <th>Start_Date</th> <th>Booked_Day(s)</th> <th>Booked_Time</th> <th>End_Date</th> </tr> </thead> <tbody> <tr> <td>8</td> <td>10</td> <td>4</td> <td>3</td> <td>30 Weeks</td> <td>27/10/2016</td> <td>Thursday</td> <td>09:30</td> <td>25/05/2017</td> </tr> <tr> <td>*</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>			ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date	8	10	4	3	30 Weeks	27/10/2016	Thursday	09:30	25/05/2017	*								
ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date																					
8	10	4	3	30 Weeks	27/10/2016	Thursday	09:30	25/05/2017																					
*																													

2.11[A][1] Screenshot D

The screenshot shows a Windows application window titled "frmScheduleTable". The main title bar says "Scheduled Lessons List". The interface includes several input fields for Student ID (13), First Name (Matthew), Surname (Jones), Contact Number (46335362), Teacher ID (2), First Name (Carrie), Surname (Gamble), Specialisation (Wood Wind), Room (2), Grade ID (3), and Grade (Advanced). There is also a "ScheduleID" field with value 25. A large grid table below shows a single row of data: ScheduleID 25, StudentID 13, TeacherID 2, LessonsPurchasedID 3, Number_Of_Weeks 30 Weeks, Start Date 01/11/2016, Booked_Day(s) Monday, Booked_Time 15:00, and End Date 30/05/2017. To the right of the grid is a sidebar with buttons for "Search for Student" (set to Adam Ackles), "previous Student Record" and "Next Student Record", "Add new Student" (green button), "Update Lesson Record" (orange button), "Remove Lesson Record" (red button), "Return to Calendar [Placeholder]" (orange button), and "Return to Main Menu" (red button).

2.11[A][2] – Grade Information**2.11[A][2] Screenshot A**

The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. On the left is the Object Explorer tree view showing various database objects like frmPrivateTuition.cs [Design], frmPrivateTuition.cs, frmSplash.cs, frmInstrument.cs [Design], frmGrade.cs, frmRoom.cs, and frmScheduleTable.cs. The central pane displays the "dbo.Grade" table data. The table has columns GradeID, GradeLevel, and Grade Fee. The data shows four rows: GradeID 1 (GradeLevel Beginner, Grade Fee 10), GradeID 2 (GradeLevel Intermediate, Grade Fee 15), GradeID 3 (GradeLevel Advanced, Grade Fee 20), and GradeID 4 (GradeLevel Diploma, Grade Fee 25). A status bar at the bottom indicates "Max Rows: 1000".

2.11[A][2] Screenshot B

Scheduled Lessons List

StudentID: <input type="text" value="6"/>	TeacherID: <input type="text" value="1"/>																											
Student Details First Name: <input type="text" value="Daniel"/> Surname: <input type="text" value="Waters"/> Contact Number: <input type="text" value="57719048"/>																												
Teacher Details TeacherID: <input type="text" value="1"/> First Name: <input type="text" value="Sandra"/> Surname: <input type="text" value="Smith"/> Specialisation: <input type="text" value="String"/> Room: <input type="text" value="1"/>																												
Grade Details GradeID: <input type="text" value="4"/> Grade: <input type="text" value="Diploma"/>																												
ScheduleID: <input type="text" value="11"/>																												
<table border="1"> <thead> <tr> <th>ScheduleID</th> <th>StudentID</th> <th>TeacherID</th> <th>LessonsPurchasedID</th> <th>Number_Of_Weeks</th> <th>Start_Date</th> <th>Booked_Day(s)</th> <th>Booked_Time</th> <th>End_Date</th> </tr> </thead> <tbody> <tr> <td>► 11</td> <td>6</td> <td>1</td> <td>3</td> <td>30 Weeks</td> <td>28/10/2016</td> <td>Monday</td> <td>09:30</td> <td>26/05/2017</td> </tr> <tr> <td>*</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>		ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date	► 11	6	1	3	30 Weeks	28/10/2016	Monday	09:30	26/05/2017	*								
ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date																				
► 11	6	1	3	30 Weeks	28/10/2016	Monday	09:30	26/05/2017																				
*																												

Actions:

- Search for Student: Adam Ackles
- previous Student Record
- Next Student Record
- Add new Student
- Update Lesson Record
- Remove Lesson Record
- Return to Calendar (Placeholder)
- Return to Main Menu

2.11[A][2] Screenshot C

Scheduled Lessons List

StudentID: <input type="text" value="10"/>	TeacherID: <input type="text" value="4"/>																											
Student Details First Name: <input type="text" value="Lorraine"/> Surname: <input type="text" value="Silvers"/> Contact Number: <input type="text" value="40027364"/>																												
Teacher Details TeacherID: <input type="text" value="4"/> First Name: <input type="text" value="Nathan"/> Surname: <input type="text" value="Jones"/> Specialisation: <input type="text" value="Percussion"/> Room: <input type="text" value="4"/>																												
Grade Details GradeID: <input type="text" value="1"/> Grade: <input type="text" value="Beginner"/>																												
ScheduleID: <input type="text" value="8"/>																												
<table border="1"> <thead> <tr> <th>ScheduleID</th> <th>StudentID</th> <th>TeacherID</th> <th>LessonsPurchasedID</th> <th>Number_Of_Weeks</th> <th>Start_Date</th> <th>Booked_Day(s)</th> <th>Booked_Time</th> <th>End_Date</th> </tr> </thead> <tbody> <tr> <td>► 8</td> <td>10</td> <td>4</td> <td>3</td> <td>30 Weeks</td> <td>27/10/2016</td> <td>Thursday</td> <td>09:30</td> <td>25/05/2017</td> </tr> <tr> <td>*</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>		ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date	► 8	10	4	3	30 Weeks	27/10/2016	Thursday	09:30	25/05/2017	*								
ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date																				
► 8	10	4	3	30 Weeks	27/10/2016	Thursday	09:30	25/05/2017																				
*																												

Actions:

- Search for Student: Adam Ackles
- previous Student Record
- Next Student Record
- Add new Student
- Update Lesson Record
- Remove Lesson Record
- Return to Calendar (Placeholder)
- Return to Main Menu

2.11[A][2] Screenshot D

The screenshot shows a Windows application window titled "frmScheduleTable". The title bar includes the application name and a logo of a guitar. The main area is titled "Scheduled Lessons List". On the left, there are three panels: "Student Details" (StudentID: 13, First Name: Matthew, Surname: Jones, Contact Number: 46335362), "Teacher Details" (TeacherID: 2, First Name: Carey, Surname: Gamble, Specialisation: Wood Wind, Room: 2), and "Grade Details" (GradeID: 3, Grade: Advanced). Below these is a "ScheduleID" field (25). To the right is a control panel with buttons for "Search for Student" (set to Adam Ackles), "Add new Student", "Update Lesson Record", "Remove Lesson Record", "Return to Calendar [Placeholder]", and "Return to Main Menu". At the bottom is a grid table:

ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date
25	13	2	3	30 Weeks	01/11/2016	Monday	15:00	30/05/2017
*								

2.11[A][3] – Teacher Information**2.11[A][3] Screenshot A**

The screenshot shows a Microsoft SQL Server Management Studio (SSMS) interface. The top navigation bar includes "dbo.Teachers [Data]" (selected), "dbo.Grade [Data]", "dbo.Students [Data]", "PurchasedLessonBundles.cs [Design]", "PurchasedLessonBundles.cs", "frmIndexMenu.cs [Design]", "frmIndexMenu.cs", and "frmGrade.cs [Design]". The main area displays a table titled "Teachers" with the following data:

TeacherID	First_Name	Surname	Address	Town	PostCode	Email_Address	Contact_Num...	Specialisation	RoomID
1	Sandra	Smith	362 Atlantic Ro...	Colecraine	BT32 3RE	ssmith362@gm...	84933740	String	1
2	Carey	Gamble	84 Colecraine Ro...	Garvagh	BT84 8IE	careygamble82...	47119403	Wood Wind	2
3	Donna	Shaw	205 Strand Ro...	Ballymoney	BT04 3RD	donashaw@gm...	49573628	Keyboard	3
4	Nathan	Jones	320 Church Stre...	Castlerock	BT56 3QB	njones542@hot...	47118394	Precussion	4
5	Garry	Gibson	102 Portrush Ro...	Colecraine	BT29 9MV	gribson920@q...	47188574	Brass	5
6	Jamie	Dome	162 Strand Ro...	Ballymoney	BT84 5BP	jdome@gmail...	48912647	Vocal	6
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

2.11[A][3] Screenshot B

Scheduled Lessons List

StudentID: 6

Student Details

- First Name: Daniel
- Surname: Waters
- Contact Number: 37719048

Teacher Details

- TeacherID: 1
- First Name: Sandra
- Surname: Smith
- Specialisation: String
- Room: 1

Grade Details

- GradeID: 4
- Grade: Diploma

ScheduleID: 11

ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date
11	6	1	3	30 Weeks	28/10/2016	Monday	09:30	26/05/2017
*								

Actions:

- Search for Student: Adam Ackles
- previous Student Record
- Next Student Record
- Add new Student
- Update Lesson Record
- Remove Lesson Record
- Return to Calendar [Placeholder]
- Return to Main Menu

2.11[A][3] Screenshot C

Scheduled Lessons List

StudentID: 10

Student Details

- First Name: Lorraine
- Surname: Silvers
- Contact Number: 40027364

Teacher Details

- TeacherID: 4
- First Name: Nathan
- Surname: Jones
- Specialisation: Precussion
- Room: 4

Grade Details

- GradeID: 1
- Grade: Beginner

ScheduleID: 8

ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date
8	10	4	3	30 Weeks	27/10/2016	Thursday	09:30	25/05/2017
*								

Actions:

- Search for Student: Adam Ackles
- previous Student Record
- Next Student Record
- Add new Student
- Update Lesson Record
- Remove Lesson Record
- Return to Calendar [Placeholder]
- Return to Main Menu

2.11[A][3] Screenshot D

The screenshot shows a Windows application window titled "Scheduled Lessons List". The interface includes:

- Student Details:** StudentID: 13, First Name: Matthew, Surname: Jones, Contact Number: 46335362.
- Teacher Details:** TeacherID: 2, First Name: Carey, Surname: Gamble, Specialisation: Wood Wind, Room: 2.
- Grade Details:** GradeID: 3, Grade: Advanced.
- ScheduleID:** 25.
- Schedule Table:**

ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date
25	13	2	3	30 Weeks	01/11/2016	Monday	15:00	30/05/2017
- Buttons:** Search for Student (with dropdown), previous Student Record, Next Student Record, Add new Student (green), Update Lesson Record (orange), Remove Lesson Record (red), Return to Calendar [Placeholder] (orange), and Return to Main Menu (red).

2.11[A][4] – Scheduled Lessons Information**2.11[A][4] Screenshot A**

The screenshot shows a database grid titled "frmLessonBundle.cs [Design]" displaying scheduled lessons. The columns are:

- ScheduleID
- StudentID
- TeacherID
- LessonsPurcha...
- Number_Of_W...
- Start_Date
- Booked_Day(s)
- Booked_Time
- End_Date

The data in the grid is as follows:

ScheduleID	StudentID	TeacherID	LessonsPurcha...	Number_Of_W...	Start_Date	Booked_Day(s)	Booked_Time	End_Date
1	7	2	1	10 Weeks	26/10/2016	Monday ... 09:30	04/01/2017	
2	7	5	2	10 Weeks	24/10/2016	Tuesday + Wed... 09:30	02/01/2017	
3	10	4	3	30 Weeks	27/10/2016	Thursday ... 09:30	25/05/2017	
4	6	1	3	30 Weeks	28/10/2016	Monday ... 09:30	26/05/2017	
5	16	1	1	5 Weeks	25/10/2016	Monday + Frid... 11:30	29/11/2016	
6	12	3	1	10 Weeks	24/10/2016	Tuesday ... 12:00	02/01/2017	
7	22	2	2	15 Weeks	28/10/2016	Thursday + Frid... 13:00	10/02/2017	
8	5	3	1	5 Weeks	25/10/2016	Monday + Tues... 14:30	29/11/2016	
9	17	4	2	15 Weeks	27/10/2016	Wednesday + T... 16:00	09/02/2017	
10	4	2	1	10 Weeks	05/01/2017	Friday ... 10:30	04/01/2017	
11	8	3	2	20 Weeks	28/10/2016	Thursday ... 11:00	17/03/2017	
12	5	2	2	20 Weeks	24/10/2016	Friday ... 13:00	13/03/2017	
13	21	3	1	5 Weeks	26/10/2016	Monday + Tues... 12:00	30/11/2016	
14	19	3	1	10 Weeks	28/10/2016	Wednesday ... 13:00	06/01/2017	
15	6	1	1	10 Weeks	26/10/2016	Thursday ... 10:00	04/01/2017	
16	10	2	2	20 Weeks	03/11/2016	Friday ... 14:00	23/03/2017	
17	8	1	2	15 Weeks	02/11/2016	Wednesday + T... 15:00	15/02/2017	
18	13	2	3	30 Weeks	01/11/2016	Monday ... 15:00	30/05/2017	
19	27	3	1	5 Weeks	03/11/2016	Tuesday + Frid... 14:00	08/12/2016	
20	6	5	2	15 Weeks	14/11/2016	Monday + Tues... 13:00	27/02/2017	
21	18	3	1	5 Weeks	15/11/2016	Wednesday + F... 13:00	20/12/2016	
22	15	3	2	20 Weeks	21/11/2016	Monday + Tues... 14:00	10/04/2017	
23	13	2	3	20 Weeks	01/09/2017	Tuesday + Wed... 13:30	01/01/2017	
24	12	5	2	15 Weeks	01/09/2017	Monday + Frid... 14:00	14/01/2017	
25	19	3	1	10 Weeks	01/09/2017	Tuesday ... 16:30	10/04/2017	
26	22	2	3	25 Weeks	01/09/2017	Friday ... 09:30	10/03/2017	
27	NULL	NULL	NULL	NULL	NULL	NULL	NULL	

2.11[A][4] Screenshot B

Scheduled Lessons List

StudentID: 6	TeacherID: 1	Search for Student: Adam Ackles																											
Student Details		previous Student Record																											
First Name: Daniel	Surname: Walters	Next Student Record																											
Contact Number: 37719048																													
Teacher Details		Add new Student																											
TeacherID: 1	First Name: Sandra	Update Lesson Record																											
Surname: Smith	Specialisation: String	Remove Lesson Record																											
Contact Number: 1																													
Grade Details		Return to Calendar [Placeholder]																											
GradeID: 4	Grade: Diploma	Return to Main Menu																											
ScheduleID: 11																													
<table border="1"> <thead> <tr> <th>ScheduleID</th> <th>StudentID</th> <th>TeacherID</th> <th>LessonsPurchasedID</th> <th>Number_of_Weeks</th> <th>Start_Date</th> <th>Booked_Day(s)</th> <th>Booked_Time</th> <th>End_Date</th> </tr> </thead> <tbody> <tr> <td>11</td> <td>6</td> <td>1</td> <td>3</td> <td>30 Weeks</td> <td>28/10/2016</td> <td>Monday</td> <td>09:30</td> <td>26/05/2017</td> </tr> <tr> <td>*</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>			ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date	11	6	1	3	30 Weeks	28/10/2016	Monday	09:30	26/05/2017	*								
ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date																					
11	6	1	3	30 Weeks	28/10/2016	Monday	09:30	26/05/2017																					
*																													

2.11[A][4] Screenshot C

Scheduled Lessons List

StudentID: 10	TeacherID: 4	Search for Student: Adam Ackles																											
Student Details		previous Student Record																											
First Name: Loraine	Surname: Silvers	Next Student Record																											
Contact Number: 40027364																													
Teacher Details		Add new Student																											
TeacherID: 4	First Name: Nathan	Update Lesson Record																											
Surname: Jones	Specialisation: Percussion	Remove Lesson Record																											
Contact Number: 4																													
Grade Details		Return to Calendar [Placeholder]																											
GradeID: 1	Grade: Beginner	Return to Main Menu																											
ScheduleID: 8																													
<table border="1"> <thead> <tr> <th>ScheduleID</th> <th>StudentID</th> <th>TeacherID</th> <th>LessonsPurchasedID</th> <th>Number_of_Weeks</th> <th>Start_Date</th> <th>Booked_Day(s)</th> <th>Booked_Time</th> <th>End_Date</th> </tr> </thead> <tbody> <tr> <td>8</td> <td>10</td> <td>4</td> <td>3</td> <td>30 Weeks</td> <td>27/10/2016</td> <td>Thursday</td> <td>09:30</td> <td>25/05/2017</td> </tr> <tr> <td>*</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>			ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date	8	10	4	3	30 Weeks	27/10/2016	Thursday	09:30	25/05/2017	*								
ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date																					
8	10	4	3	30 Weeks	27/10/2016	Thursday	09:30	25/05/2017																					
*																													

2.11[A][4] Screenshot D

The screenshot shows the 'Scheduled Lessons List' application window. On the left, there are three input boxes: 'StudentID: 13', 'TeacherID: 2', and 'ScheduleID: 25'. Below these are three groups of details: 'Student Details' (First Name: Matthew, Surname: Jones, Contact Number: 46335362), 'Teacher Details' (First Name: Carey, Surname: Gamble, Specialisation: Wood Wind, Room: 2), and 'Grade Details' (GradeID: 3, Grade: Advanced). To the right is a large table containing one row of data: ScheduleID 25, StudentID 13, TeacherID 2, LessonsPurchasedID 3, Number_of_Weeks 30 Weeks, Start Date 01/11/2016, Booked_Day(s) Monday, Booked_Time 15:00, and End Date 30/05/2017. On the far right, a sidebar contains buttons for 'Search for Students' (set to Adam Ackles), 'previous Student Record' (disabled), 'Next Student Record' (disabled), 'Add new Student' (green), 'Update Lesson Record' (orange), 'Remove Lesson Record' (red), 'Return to Calendar [Placeholder]' (disabled), and 'Return to Main Menu' (red).

Process [B]) Record Navigation**2.11[B][1] – Student Search****2.11[B][1] Screenshot A**

This screenshot is identical to Screenshot D, showing the same application interface. The 'Search for Students' dropdown is now populated with a list of names: Adam Ackles, Alan Blue, Anthony Howard, Ben Kite, Courtney Doherty, Daniel Waters, Diana James, Duke Evans, Eliza Cobb, Gerard Mcdead, Jenson Tweed, Kayla Black, Kirsteen Dickenson, Linda Daniels, Linda Simmons, Linda Spears, Loraine Silvers, Louise Jordan, Lucy Dyson, Matthew Jones, Patrick Roberts, Reginald Stevenson, Samantha Mcfarlane, and Trevor Simmons. The rest of the interface, including the student and teacher details, grade details, and the schedule table, remains the same.

2.11[B][1] Screenshot B

Scheduled Lessons List

StudentID: 10	TeacherID: 4	Search for Student: Loraine Silvers																											
Student Details		previous Student Record																											
First Name: Loraine	Surname: Silvers	Next Student Record																											
Contact Number: 40027364																													
Teacher Details																													
GradeID: 1	Grade: Beginner	Add new Student																											
Grade Details		Update Lesson Record																											
ScheduleID: 8	Remove Lesson Record																												
<table border="1"> <thead> <tr> <th>ScheduleID</th> <th>StudentID</th> <th>TeacherID</th> <th>LessonsPurchasedID</th> <th>Number_Of_Weeks</th> <th>Start_Date</th> <th>Booked_Day(s)</th> <th>Booked_Time</th> <th>End_Date</th> </tr> </thead> <tbody> <tr> <td>8</td> <td>10</td> <td>4</td> <td>3</td> <td>30 Weeks</td> <td>27/10/2016</td> <td>Thursday</td> <td>09:30</td> <td>25/05/2017</td> </tr> <tr> <td>*</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>			ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date	8	10	4	3	30 Weeks	27/10/2016	Thursday	09:30	25/05/2017	*								
ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date																					
8	10	4	3	30 Weeks	27/10/2016	Thursday	09:30	25/05/2017																					
*																													

2.11[B][2] – Next Student**2.11[B][2] Screenshot A**

Scheduled Lessons List

StudentID: 10	TeacherID: 4	Search for Student: Loraine Silvers																											
Student Details		previous Student Record																											
First Name: Loraine	Surname: Silvers	Next Student Record																											
Contact Number: 40027364																													
Teacher Details																													
GradeID: 1	Grade: Beginner	Add new Student																											
Grade Details		Update Lesson Record																											
ScheduleID: 8	Remove Lesson Record																												
<table border="1"> <thead> <tr> <th>ScheduleID</th> <th>StudentID</th> <th>TeacherID</th> <th>LessonsPurchasedID</th> <th>Number_Of_Weeks</th> <th>Start_Date</th> <th>Booked_Day(s)</th> <th>Booked_Time</th> <th>End_Date</th> </tr> </thead> <tbody> <tr> <td>8</td> <td>10</td> <td>4</td> <td>3</td> <td>30 Weeks</td> <td>27/10/2016</td> <td>Thursday</td> <td>09:30</td> <td>25/05/2017</td> </tr> <tr> <td>*</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>			ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date	8	10	4	3	30 Weeks	27/10/2016	Thursday	09:30	25/05/2017	*								
ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date																					
8	10	4	3	30 Weeks	27/10/2016	Thursday	09:30	25/05/2017																					
*																													

2.11[B][2] Screenshot B

The screenshot shows the 'Scheduled Lessons List' application window. On the left, there are two input panels: 'Student Details' (StudentID: 11, First Name: Courtney, Surname: Doherty, Contact Number: 47326678) and 'Teacher Details' (TeacherID: [empty], First Name: [empty], Surname: [empty], Specialisation: [empty], Room: [empty]). Below these are 'Grade Details' (GradeID: 1, Grade: Beginner). A 'ScheduleID:' input field is present. To the right is a large table titled 'ScheduleTable' with columns: ScheduleID, StudentID, TeacherID, LessonsPurchasedID, Number_of_Weeks, Start_Date, Booked_Day(s), Booked_Time, and End_Date. The table has one row with values: * (11, [empty], [empty], [empty], [empty], [empty], [empty], [empty], [empty]). At the top right, a search bar shows 'Search for Student: Loraine Silvers'. Below it are buttons for 'Add new Student', 'Update Lesson Record', 'Remove Lesson Record', 'Return to Calendar [Placeholder]', and 'Return to Main Menu'.

2.11[B][3] – Previous Student**2.11[B][3] Screenshot A**

This screenshot is identical to Screenshot B, showing the 'Scheduled Lessons List' application. It displays the same student and teacher details, grade details, and schedule table. The search bar at the top right also shows 'Search for Student: Loraine Silvers'. The interface includes buttons for managing student and lesson records, as well as links to the calendar and main menu.

2.11[B][3] Screenshot B

Scheduled Lessons List

StudentID: 10	TeacherID: 4	Search for Student: Loraine Silvers
Student Details		previous Student Record
First Name: Loraine	First Name: Nathan	Next Student Record
Surname: Silvers	Surname: Jones	
Contact Number: 40027364	Specialisation: Percussion	Add new Student
	Room: 4	Update Lesson Record
Grade Details		Remove Lesson Record
GradeID: 1	Grade: Beginner	
ScheduleID: 8	Return to Calendar [Placeholder]	
Return to Main Menu		

Schedule Details:

ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date
8	10	4	3	30 Weeks	27/10/2016	Thursday	09:30	25/05/2017
*								

2.11[B][4] – First Student**2.11[B][4] Screenshot A**

Scheduled Lessons List

StudentID: 10	TeacherID: 4	Search for Student: Loraine Silvers
Student Details		previous Student Record
First Name: Loraine	First Name: Nathan	Next Student Record
Surname: Silvers	Surname: Jones	
Contact Number: 40027364	Specialisation: Percussion	Add new Student
	Room: 4	Update Lesson Record
Grade Details		Remove Lesson Record
GradeID: 1	Grade: Beginner	
ScheduleID: 8	Return to Calendar [Placeholder]	
Return to Main Menu		

Schedule Details:

ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date
8	10	4	3	30 Weeks	27/10/2016	Thursday	09:30	25/05/2017
*								

2.11[B][4] Screenshot B

Scheduled Lessons List

The screenshot shows the 'Scheduled Lessons List' application window. On the left, there are input fields for Student ID (4), Student Details (First Name: Linda, Surname: Simmons, Contact Number: 70359371), Teacher Details (TeacherID: [empty], First Name: [empty], Surname: [empty], Specialisation: [empty], Room: [empty]), Grade Details (GradeID: 2, Grade: Intermediate), and a ScheduleID input field. On the right, a sidebar contains buttons for 'Search for Student' (Lorane Slivers), 'previous Student Record' and 'Next Student Record', and links for 'Add new Student', 'Update Lesson Record', 'Remove Lesson Record', 'Return to Calendar [Placeholder]', and 'Return to Main Menu'.

ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date
*								

2.11[B][5] – Last Student**2.11[B][5] Screenshot A**

Scheduled Lessons List

The screenshot shows the 'Scheduled Lessons List' application window. It is identical to Screenshot B, displaying the same student and teacher details, grade details, and schedule table. The sidebar on the right also contains the same buttons and links.

ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date
*								

2.11[C][5] Screenshot B

Scheduled Lessons List

StudentID: <input type="text" value="27"/>	TeacherID: <input type="text"/>																		
Student Details First Name: <input type="text" value="Frederica"/> Surname: <input type="text" value="Cobb"/> Contact Number: <input type="text" value="70329356"/>																			
Teacher Details TeacherID: <input type="text"/> First Name: <input type="text"/> Surname: <input type="text"/> Specialisation: <input type="text"/> Room: <input type="text"/>																			
Grade Details GradeID: <input type="text" value="3"/> Grade: <input type="text" value="Advanced"/>																			
ScheduleID: <input type="text"/>																			
<table border="1"> <thead> <tr> <th>ScheduleID</th> <th>StudentID</th> <th>TeacherID</th> <th>LessonsPurchasedID</th> <th>Number_Of_Weeks</th> <th>Start_Date</th> <th>Booked_Day(s)</th> <th>Booked_Time</th> <th>End_Date</th> </tr> </thead> <tbody> <tr> <td>*</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>		ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date	*								
ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date											
*																			

Search for Student:
 << previous Student Record Next Student Record >>
 Add new Student Update Lesson Record Remove Lesson Record
 Return to Calendar [Placeholder] Return to Main Menu

Process [CJ] Form Navigation**2.11[C][1] – Add New Scheduled Lesson Record****2.11[C][1] Screenshot A**

Scheduled Lessons List

StudentID: <input type="text" value="27"/>	TeacherID: <input type="text"/>																		
Student Details First Name: <input type="text" value="Frederica"/> Surname: <input type="text" value="Cobb"/> Contact Number: <input type="text" value="70329356"/>																			
Teacher Details TeacherID: <input type="text"/> First Name: <input type="text"/> Surname: <input type="text"/> Specialisation: <input type="text"/> Room: <input type="text"/>																			
Grade Details GradeID: <input type="text" value="3"/> Grade: <input type="text" value="Advanced"/>																			
ScheduleID: <input type="text"/>																			
<table border="1"> <thead> <tr> <th>ScheduleID</th> <th>StudentID</th> <th>TeacherID</th> <th>LessonsPurchasedID</th> <th>Number_Of_Weeks</th> <th>Start_Date</th> <th>Booked_Day(s)</th> <th>Booked_Time</th> <th>End_Date</th> </tr> </thead> <tbody> <tr> <td>*</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>		ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date	*								
ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date											
*																			

Search for Student:
 << previous Student Record Next Student Record >>
 Add new Student Update Lesson Record Remove Lesson Record
 Return to Calendar [Placeholder] Return to Main Menu

2.11[C][1] Screenshot B

Schedule New Lesson

Student ID: 4 | Linda Simmons

Teacher ID: 1 | Sandra Smith

Lesson Bundle ID:

No. of weeks: 5 Weeks

Start Date:

Booked Time: 13:00

Booked Day(s):

[Schedule New Lesson](#) [Return to Scheduled Lessons Table](#)

2.11[C][2] – Update Scheduled Lesson Record (No Record Selected)**2.11[C][2] Screenshot A**

Scheduled Lessons List

StudentID: 27

Student Details

First Name: Frederica

Surname: Cobb

Contact Number: 70329356

Teacher Details

TeacherID:

First Name:

Surname:

Specialisation:

Room:

Grade Details

GradeID: 3

Grade: Advanced

ScheduleID:

Search for Student: Adam Ackles

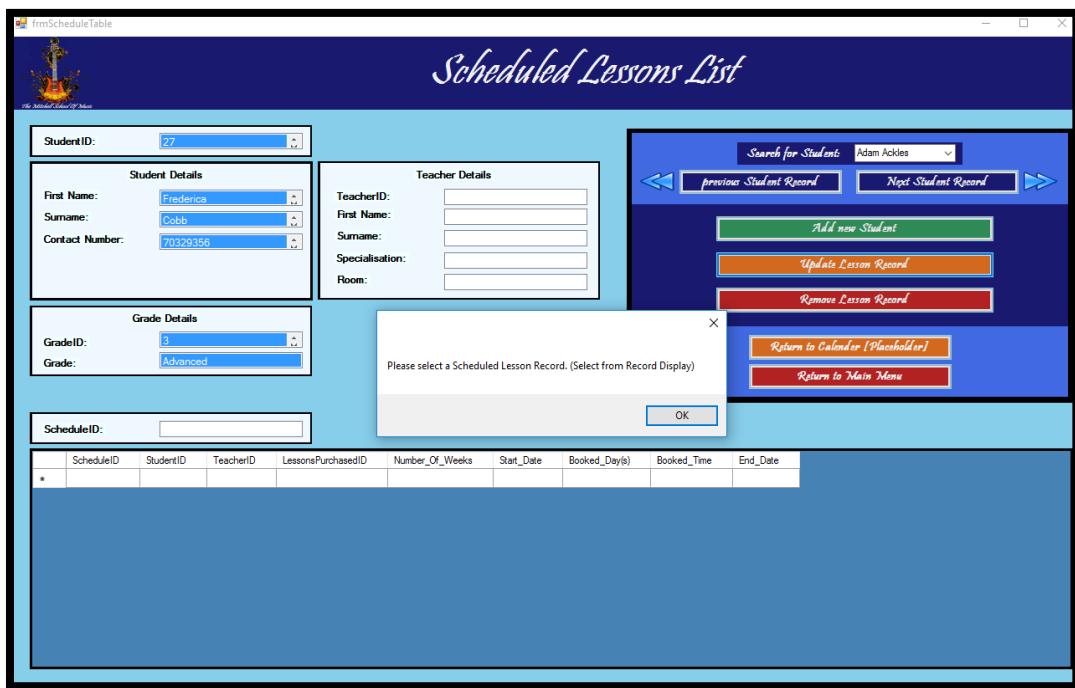
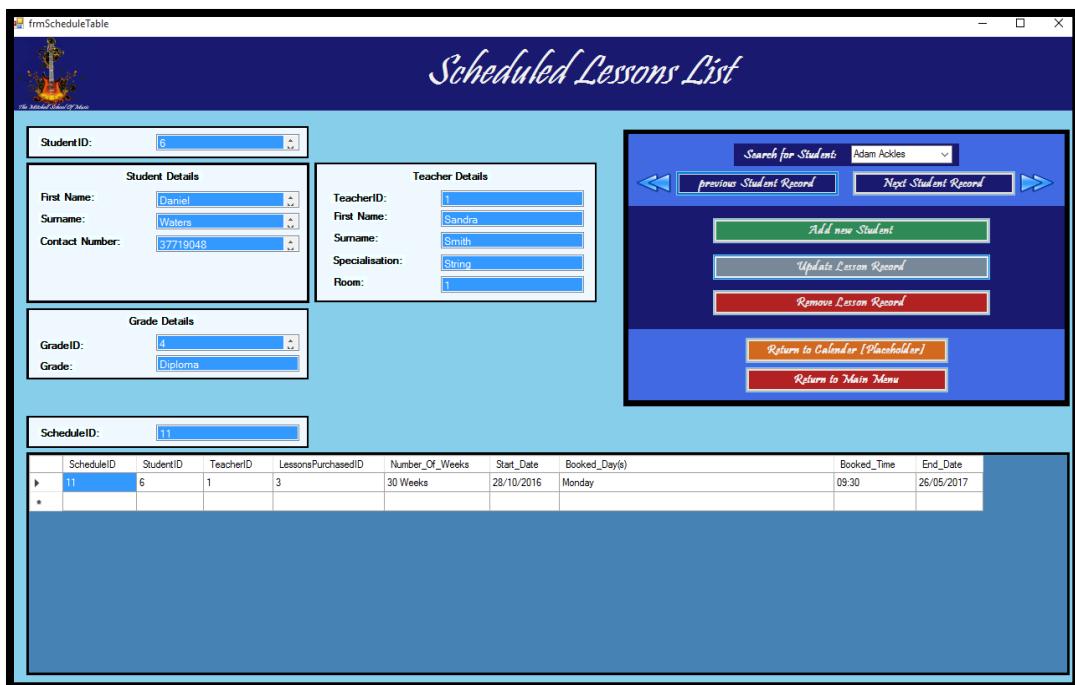
Add new Student

Update Lesson Record

Remove Lesson Record

Return to Calendar [Placeholder]

Return to Main Menu

2.11[C][2] Screenshot B**2.11[C][3] – Update Scheduled Lesson Record (Record Selected)****2.11[C][3] Screenshot A**

2.11[C][3] Screenshot B

Schedule New Lesson

Schedule ID: 11

Student ID: 4 | Linda Simmons

Teacher ID: 1 | Sandra Smith

Lesson Bundle ID:

No. of weeks: 5 Weeks

Start Date: 28/10/2016 00:00:00

Booked Time: 13:00

Booked Day(s): Monday

End Date: 26/05/2017 00:00:00

[Schedule New Lesson](#) [Return to Scheduled Lessons Table](#)

2.11[C][4] – Delete Scheduled Lesson Record [User Logged In] (No Record Selected)**2.11[C][4] Screenshot A**

Scheduled Lessons List

Student ID: 4

Student Details

First Name: Linda

Surname: Simmons

Contact Number: 70359371

Teacher Details

TeacherID: 1

First Name: Sandra

Surname: Smith

Specialisation: Room:

Grade Details

GradeID: 2

Grade: Intermediate

ScheduleID:

Search for Student: Adam Ackles

Add new Student

Update Lesson Record

Remove Lesson Record

Return to Calendar [Placeholder]

Return to Main Menu

2.11[C][4] Screenshot B

```

private void bttnRemove_Click(object sender, EventArgs e)
{
    // Set associated field.
    GlobalVariables.PreviousForm = "ScheduledLessonTable";
    // To remove a record, the user needs to login.
    if (GlobalVariables.UberLoggedIn == false)
    {
        // If no user is currently logged in, load login form.
        frmUserLogin login = new frmUserLogin();
        login.Show();
        this.Hide();
    }
    else
    {
        // If a user is logged in, load confirmation page.
        GlobalVariables.TableID = Convert.ToInt32(lbScheduledLessons.Text);
        GlobalVariables.FieldName = lbFieldName.Text + " + " + lbSurname.Text;
        frmConfirmation Confirmation = new frmConfirmation();
        Confirmation.Show();
    }
}

// STORE STUDENT INFORMATION
public void StoreAllStudentIDs()
{
    // This function will be used to store various pieces of information from the schedule Table.
    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        connection.Open();
        using (SqlCommand Command = new SqlCommand("SELECT * FROM Students"))
        {
            DataReader = Command.ExecuteReader();
            while (DataReader.Read())
            {
                // For each record, store the corresponding piece of information.
                AllStudent_ID.Add(DataReader.GetInt32(0));
                AllStudent_Firstname.Add(DataReader.GetString(1));
                AllStudent_Surname.Add(DataReader.GetString(2));
            }
        }
        connection.Close();
    }
}

// FILL SEARCH BOX
private void Fill_StudentSearchBox()
{
}

```

FormatException was unhandled
An unhandled exception of type 'System.FormatException' occurred in mscoreib.dll
Additional information: Input string was not in a correct format.

Troubleshooting tips:
When converting a string to DateTime, parse the string to take the date before putting each variable into the DateTime object.
Make sure your method arguments are in the right format.
Get general help for this exception.

Exception settings:
Break when this exception type is thrown

Actions:
View Detail...
Copy exception detail to the clipboard
Open exception settings

Activate Windows

2.11[C][4][Corrective Action] – Delete Scheduled Lesson Record [User Logged In] (No Record Selected)**2.11[C][Corrective Action] Screenshot A**

```

private void bttnRemove_Click(object sender, EventArgs e)
{
    // Set associated field.
    GlobalVariables.PreviousForm = "ScheduledLessonTable";
    // To remove a record, the user needs to login.
    if (GlobalVariables.UberLoggedIn == false)
    {
        // If no user is currently logged in, load login form.
        frmUserLogin login = new frmUserLogin();
        login.Show();
        this.Hide();
    }
    else
    {
        if (lbScheduledLessons.Text != null && !string.IsNullOrWhiteSpace(lbScheduledLessons.Text))
        {
            // If a user is logged in, load confirmation page.
            GlobalVariables.TableID = Convert.ToInt32(lbScheduledLessons.Text);
            GlobalVariables.FieldName = lbFieldName.Text + " + " + lbSurname.Text;

            frmConfirmation Confirmation = new frmConfirmation();
            Confirmation.Show();
        }
        else
        {
            MessageBox.Show("Please select a scheduled Lesson Record from the menu below. If none are present, then this student has no scheduled lessons.");
        }
    }
}

// STORE STUDENT INFORMATION
public void StoreAllStudentIDs()
{
    // This function will be used to store various pieces of information from the schedule Table. This will be used to complete the schedule
    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        connection.Open();
        using (SqlCommand Command = new SqlCommand("SELECT * FROM Students", connection))
        {
            DataReader = Command.ExecuteReader();
        }
    }
}

```

2.11[C][Corrective Action] Screenshot B

Scheduled Lessons List

ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date
*	4							

2.11[C][Corrective Action] Screenshot C

Scheduled Lessons List

ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date
*	4							

Please select a scheduled Lesson Record from the menu below. If none are present, then this student has no scheduled lessons.

OK

2.11[C][5] – Delete Scheduled Lesson Record [User Logged In] (Record Selected)**2.11[C][5] Screenshot A**

Scheduled Lessons List

StudentID: <input type="text" value="6"/>	TeacherID: <input type="text" value="1"/>	Search for Student: <input type="text" value="Adam Ackles"/>																											
First Name: <input type="text" value="Daniel"/>	First Name: <input type="text" value="Sandra"/>	previous Student Record																											
Surname: <input type="text" value="Waters"/>	Surname: <input type="text" value="Smith"/>	Next Student Record																											
Contact Number: <input type="text" value="37719048"/>	Specialisation: <input type="text" value="String"/>																												
GradeID: <input type="text" value="4"/>	Room: <input type="text" value="1"/>																												
Grade: <input type="text" value="Diploma"/>																													
ScheduleID: <input type="text" value="11"/>																													
<table border="1"> <thead> <tr> <th>ScheduleID</th> <th>StudentID</th> <th>TeacherID</th> <th>LessonsPurchasedID</th> <th>Number_Of_Weeks</th> <th>Start_Date</th> <th>Booked_Day(s)</th> <th>Booked_Time</th> <th>End_Date</th> </tr> </thead> <tbody> <tr> <td>11</td> <td>6</td> <td>1</td> <td>3</td> <td>30 Weeks</td> <td>28/10/2016</td> <td>Monday</td> <td>09:30</td> <td>26/05/2017</td> </tr> <tr> <td>*</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>			ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date	11	6	1	3	30 Weeks	28/10/2016	Monday	09:30	26/05/2017	*								
ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date																					
11	6	1	3	30 Weeks	28/10/2016	Monday	09:30	26/05/2017																					
*																													

2.11[C][5] Screenshot B

Scheduled Lessons List

StudentID: <input type="text" value="6"/>	TeacherID: <input type="text" value="1"/>	Search for Student: <input type="text" value="Adam Ackles"/>																											
First Name: <input type="text" value="Daniel"/>	First Name: <input type="text" value="Sandra"/>	previous Student Record																											
Surname: <input type="text" value="Waters"/>	Surname: <input type="text" value="Smith"/>	Next Student Record																											
Contact Number: <input type="text" value="37719048"/>	Specialisation: <input type="text" value="String"/>																												
GradeID: <input type="text" value="4"/>	Room: <input type="text" value="1"/>																												
Grade: <input type="text" value="Diploma"/>																													
ScheduleID: <input type="text" value="11"/>																													
<table border="1"> <thead> <tr> <th>ScheduleID</th> <th>StudentID</th> <th>TeacherID</th> <th>LessonsPurchasedID</th> <th>Number_Of_Weeks</th> <th>Start_Date</th> <th>Booked_Day(s)</th> <th>Booked_Time</th> <th>End_Date</th> </tr> </thead> <tbody> <tr> <td>11</td> <td>6</td> <td>1</td> <td>3</td> <td>30 Weeks</td> <td>28/10/2016</td> <td>Monday</td> <td>09:30</td> <td>26/05/2017</td> </tr> <tr> <td>*</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>			ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date	11	6	1	3	30 Weeks	28/10/2016	Monday	09:30	26/05/2017	*								
ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date																					
11	6	1	3	30 Weeks	28/10/2016	Monday	09:30	26/05/2017																					
*																													

Delete Confirmation

Scheduled Lesson ID:
 Student Name:

2.11[C][6] – Delete Scheduled Lesson Record [No User Logged In]**2.11[C][6] Screenshot A**

Scheduled Lessons List

StudentID: <input type="text" value="4"/>	TeacherID: <input type="text"/>																			
Student Details First Name: <input type="text" value="Linda"/> Surname: <input type="text" value="Simmons"/> Contact Number: <input type="text" value="70359371"/>		Teacher Details First Name: <input type="text"/> Surname: <input type="text"/> Specialisation: <input type="text"/> Room: <input type="text"/>																		
Grade Details GradeID: <input type="text" value="1"/> Grade: <input type="text" value="Beginner"/>		Search for Student: Adam Ackles previous Student Record Next Student Record Add new Student Update Lesson Record Remove Lesson Record Return to Calendar Placeholder Return to Main Menu																		
ScheduleID: <input type="text"/> <table border="1"> <thead> <tr> <th>ScheduleID</th> <th>StudentID</th> <th>TeacherID</th> <th>LessonsPurchasedID</th> <th>Number_of_Weeks</th> <th>Start_Date</th> <th>Booked_Day(s)</th> <th>Booked_Time</th> <th>End_Date</th> </tr> </thead> <tbody> <tr> <td>*</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>			ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date	*								
ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date												
*																				

2.11[C][6] Screenshot B

Admin Login

User Name:	<input type="text"/>
Password:	<input type="password"/>
Return	Log In

2.11[C][7] – Return To Main Menu**2.11[C][7] Screenshot A**

Scheduled Lessons List

Student ID:	4
Student Details	
First Name:	Linda
Surname:	Simmons
Contact Number:	70359371
Teacher Details	
TeacherID:	
First Name:	
Surname:	
Specialisation:	
Room:	
Grade Details	
GradeID:	2
Grade:	Intermediate
ScheduleID:	

Search for Student: Adam Ackles

Buttons:

- previous Student Record
- Next Student Record
- Add new Student
- Update Lesson Record
- Remove Lesson Record
- Return to Calendar [Placeholder]
- Return to Main Menu

2.11[C][7] Screenshot B

Mitchell School of Music
-Private Tuition-

Main Menu

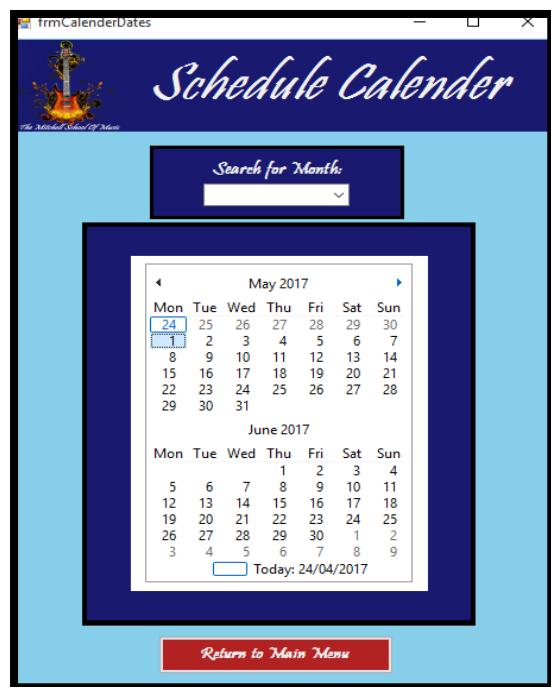
Record Tables

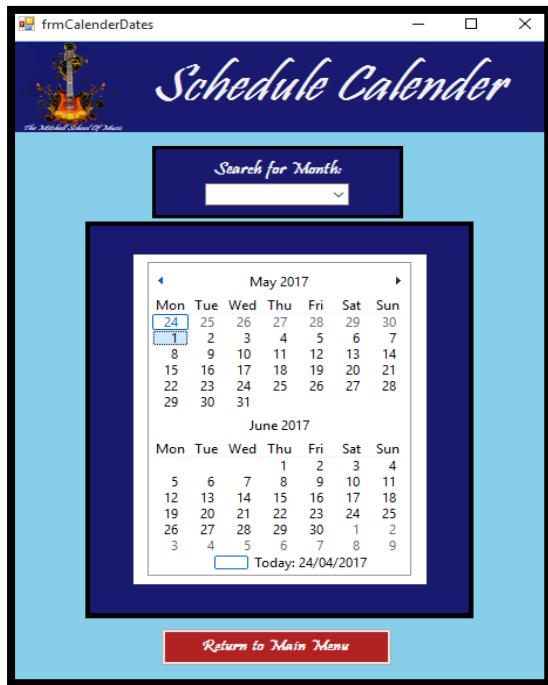
- [View Student Table](#)
- [View Teacher Table](#)
- [View Instrument Table](#)
- [View Room Table](#)
- [View Grade Table](#)
- [View Lesson Bundle Table](#)

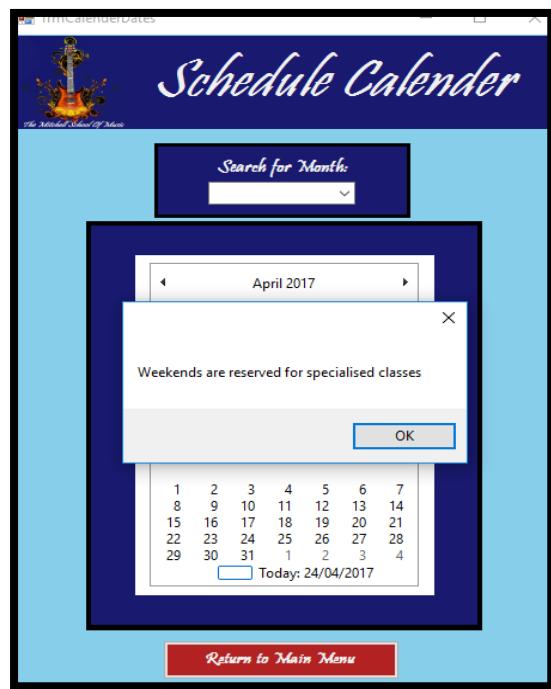
Upcoming Scheduled Lessons

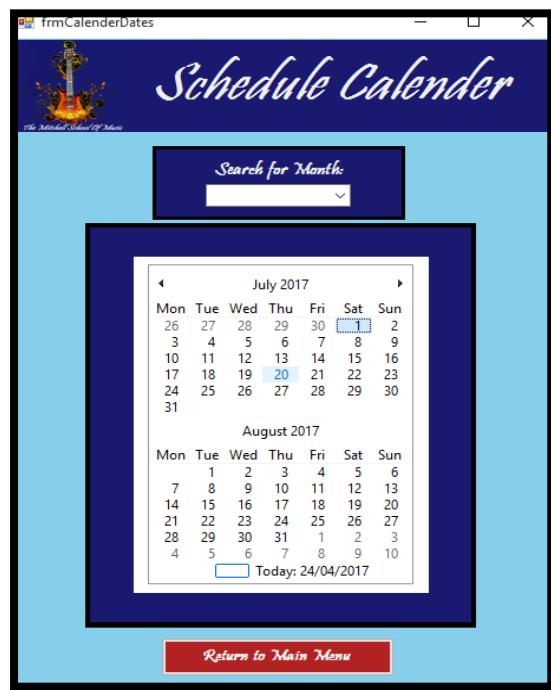
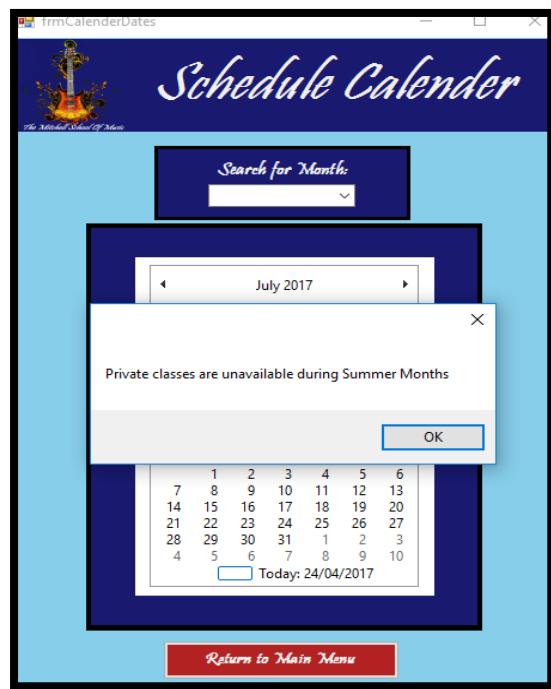
- [View Schedule Calendar](#)
- [View Scheduled Lessons by student](#)
- [View Purchased Lessons](#)
- [View Archived Scheduled Lessons](#)

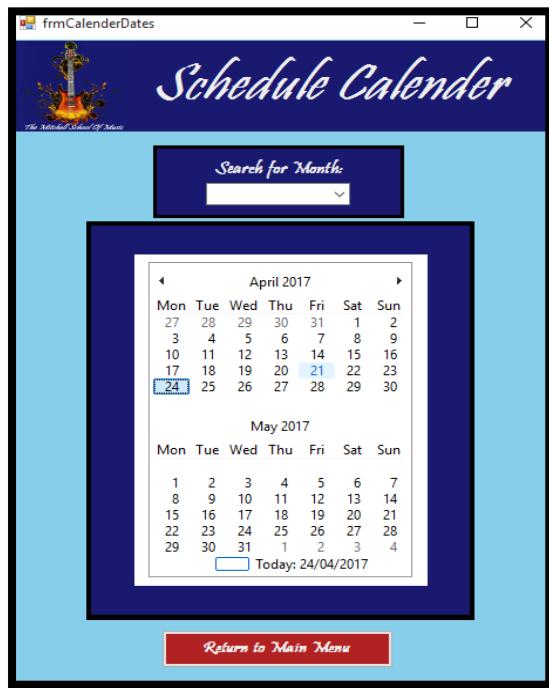
Current User: [Logout](#)

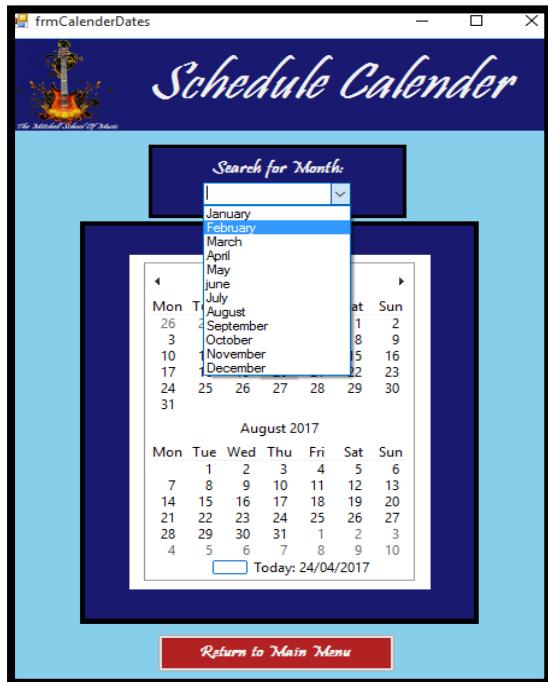
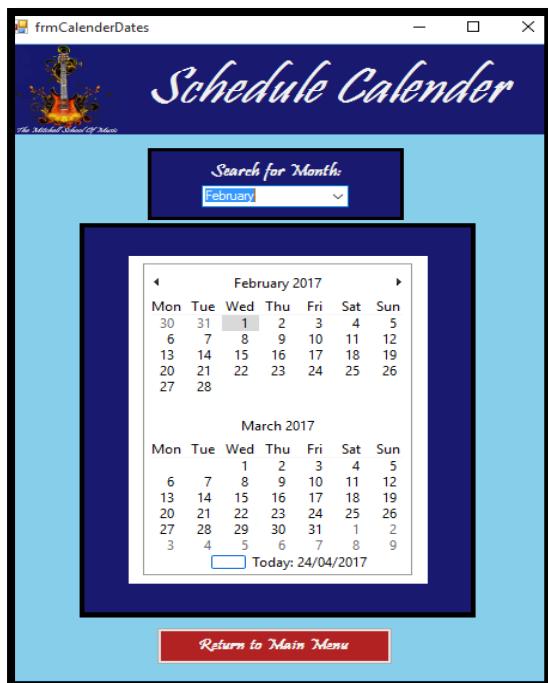
2.14) Schedule Calendar – Dates***Process [A] Calendar******2.14[A][1] – Calendar Navigation {Select Arrows}******2.14[A][1] Screenshot A******2.14[A][1] Screenshot B***

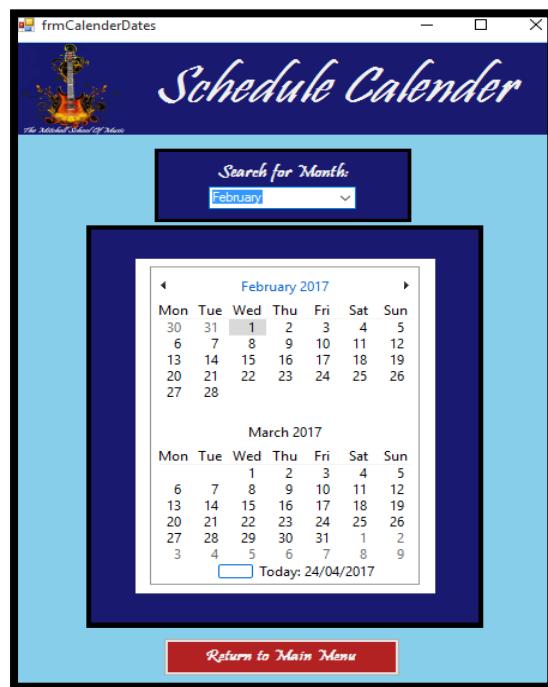
2.14[A][1] Screenshot C**2.14[A][1] Screenshot D**

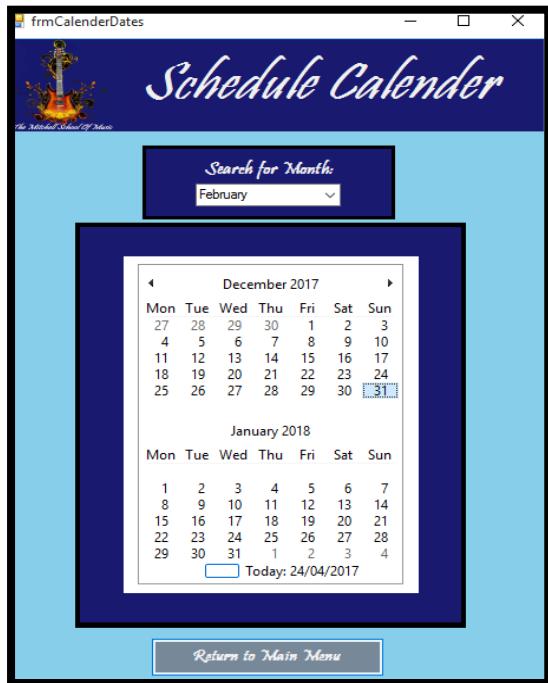
2.14[A][2] – Calendar Error {Select Weekend}**2.14[A][2] Screenshot A****2.14[A][2] Screenshot B**

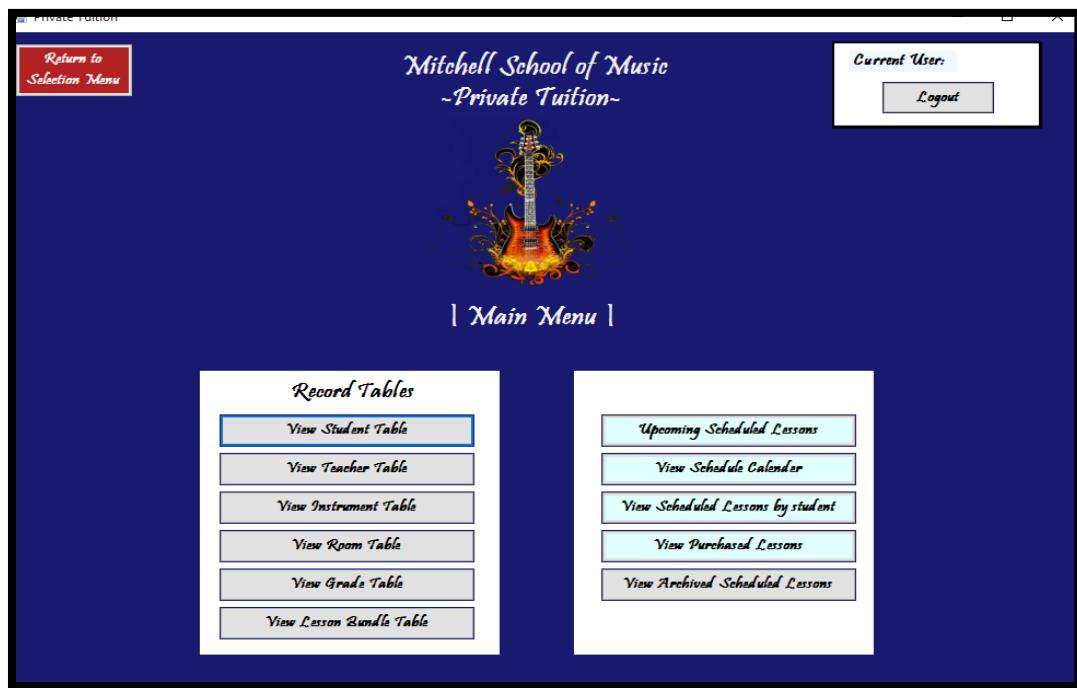
2.14[A][3] – Calendar Error {Select Summer Date}**2.14[A][3] Screenshot A****2.14[A][3] Screenshot B**

2.14[A][4] – Calendar {Select Valid Date}**2.14[A][4] Screenshot A****2.14[A][4] Screenshot B**

*Process [B]) Search Function***2.14[B][1] – Search by Month****2.14[A][1] Screenshot A****2.14[A][1] Screenshot B**

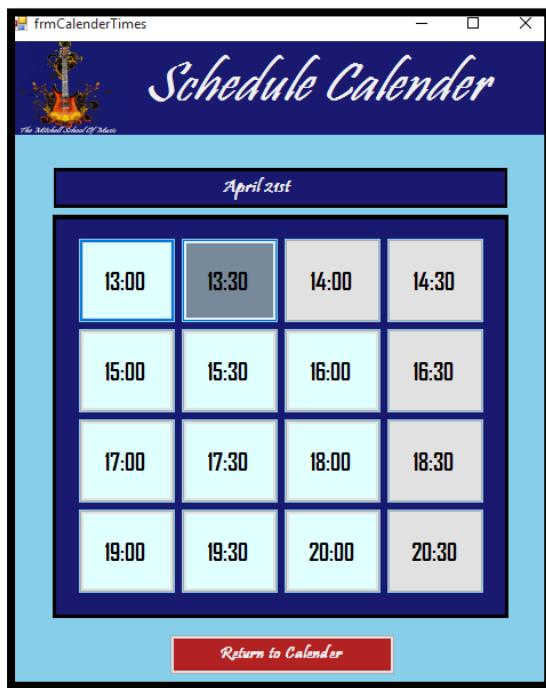
2.14[B][2] – Calendar Navigation {Select Date Display}**2.14[A][2] Screenshot A****2.14[A][2] Screenshot B**

2.14[A][2] Screenshot C**Process [C]) Form Navigation****2.14[C][1] – Return To Main Menu****2.14[A][1] Screenshot A**

2.14[A][1] Screenshot B**2.15) Schedule Calendar – Times**

Process [A]) Form Display

2.15[A][1] – Date Display**2.15[A][1] Screenshot A**

*Process [B]) Time Selection***2.15[B][1] – Time Based Buttons****2.15[A][1] Screenshot A****2.15[A][1] Screenshot B**

```

    inference
    public void ScheduledLessons()
    {
        using (connection = new SqlConnection(connectionString))
        {
            connection.Open();
            using (SqlCommand command = new SqlCommand("SELECT * FROM Scheduled_Lessons", connection))
            {
                DataReader = command.ExecuteReader();
                while (DataReader.Read())
                {
                    ScheduleID.Add(DataReader.GetInt32(0));
                    StudentID_Schedule.Add(DataReader.GetInt32(1));
                    ScheduleDate.Add(DataReader.GetDateTime(5));
                    TeacherID_Schedule.Add(DataReader.GetInt32(2));
                    DesiredDisplay.Add("false");
                    ScheduledTime.Add(DataReader.GetTimeSpan(7));
                }
            }
        }
    }

    inference
    public void StudentDetails()
    {
        using (connection = new SqlConnection(connectionString))
        {
            connection.Open();
            using (SqlCommand command2 = new SqlCommand("SELECT * FROM Students", connection))
            {
                DataReader = command2.ExecuteReader();
                while (DataReader.Read())
                {
                    StudentID_Student.Add(DataReader.GetInt32(0));
                    StudentFirstName.Add(DataReader.GetString(1));
                    StudentLastName.Add(DataReader.GetString(2));
                    Instrument_ID_Students.Add(DataReader.GetInt32(1));
                }
            }
        }
    }

    inference
    public void TeacherDetails()
    {
    }

```

An exception dialog box is displayed, showing the error message: "InvalidCastException was unhandled". It provides troubleshooting tips, including "Make sure the source type is convertible to the destination type." and "When casting from a number, the value must be a number less than infinity." The "Actions" section includes "View Detail...", "Copy exception detail to the clipboard", and "Open exception settings".

2.15[B][1][Corrective Action] – Time Based Buttons**2.15[A][Corrective Action] Screenshot A**

```

using System;
using System.Windows.Forms;
using System.Data.SqlClient;

public partial class frmPrivateTuition : Form
{
    private void btnReturn_Click(object sender, EventArgs e)
    {
        GlobalVariables.SelectedTime = "";
        GlobalVariables.SelectedDay = 0;
        GlobalVariables.SelectedMonth = 0;
        GlobalVariables.SelectedYearString = "";
        GlobalVariables.SelectedYear = 0;
        GlobalVariables.SelectedScheduleDate = "00/00/0000";

        A = 0;
        B = 0;

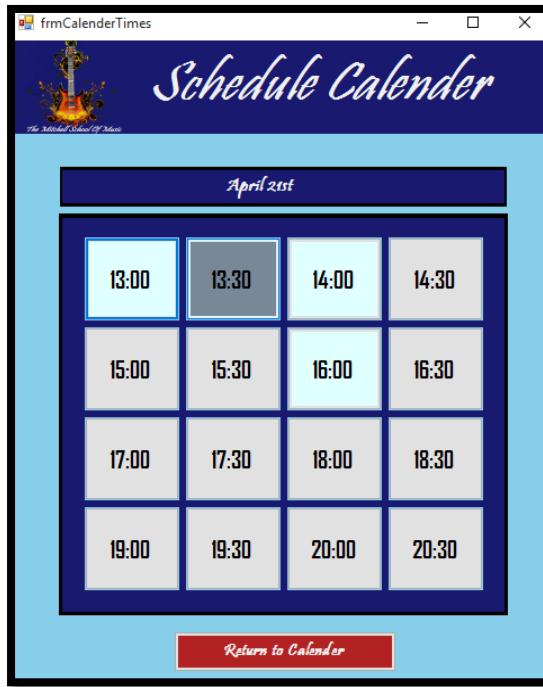
        frmPrivateTuition Main = new frmPrivateTuition();
        Main.Show();
        this.Hide();
    }

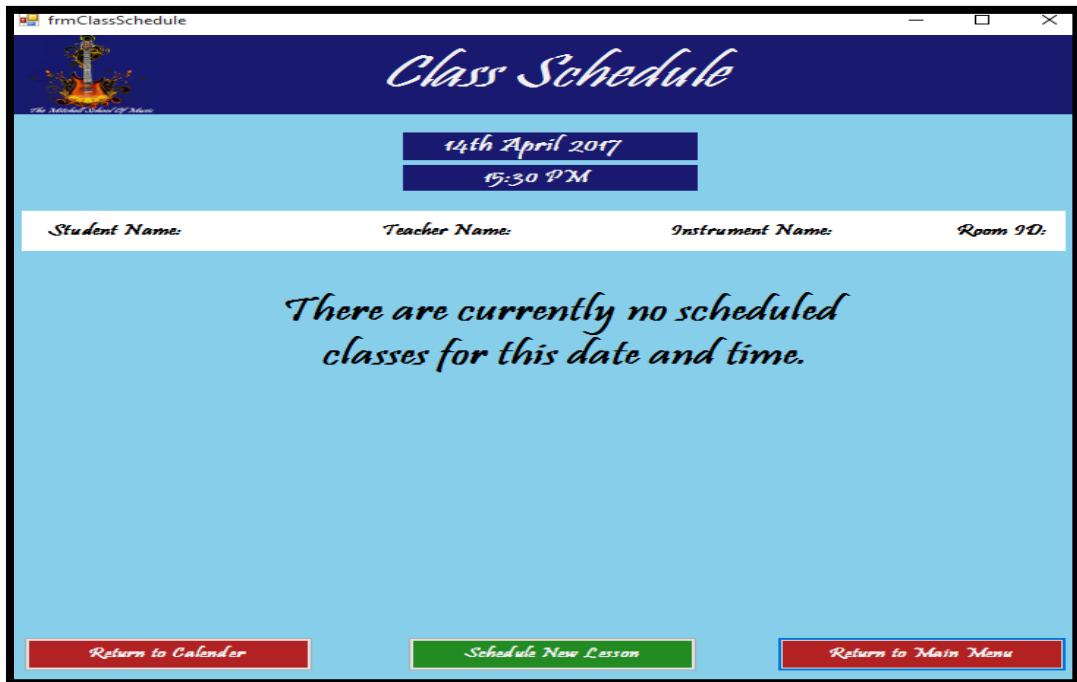
    public void ScheduledLessons()
    {
        using (connection = new SqlConnection(connectionString))
        {
            connection.Open();
            using (SqlCommand Command = new SqlCommand("SELECT * FROM Scheduled_Lessons", connection))
            {

                DataReader = Command.ExecuteReader();
                while (DataReader.Read())
                {
                    ScheduleList.Add(DataReader.GetInt32(0));
                    StudentID_Schedule.Add(DataReader.GetInt32(1));
                    ScheduleID_Schedule.Add(DataReader.GetDateTime(2));
                    TestID_Schedule.Add(DataReader.GetInt32(3));
                    TestScheduleDisplay.Add(DataReader.GetString(4));
                    ScheduleTime.Add(DataReader.GetString(5));
                }
            }
        }
    }

    public void StudentDetails()
    {
        using (connection = new SqlConnection(connectionString))
        {
            connection.Open();
            using (SqlCommand Command2 = new SqlCommand("SELECT * FROM Students", connection))
            {

```

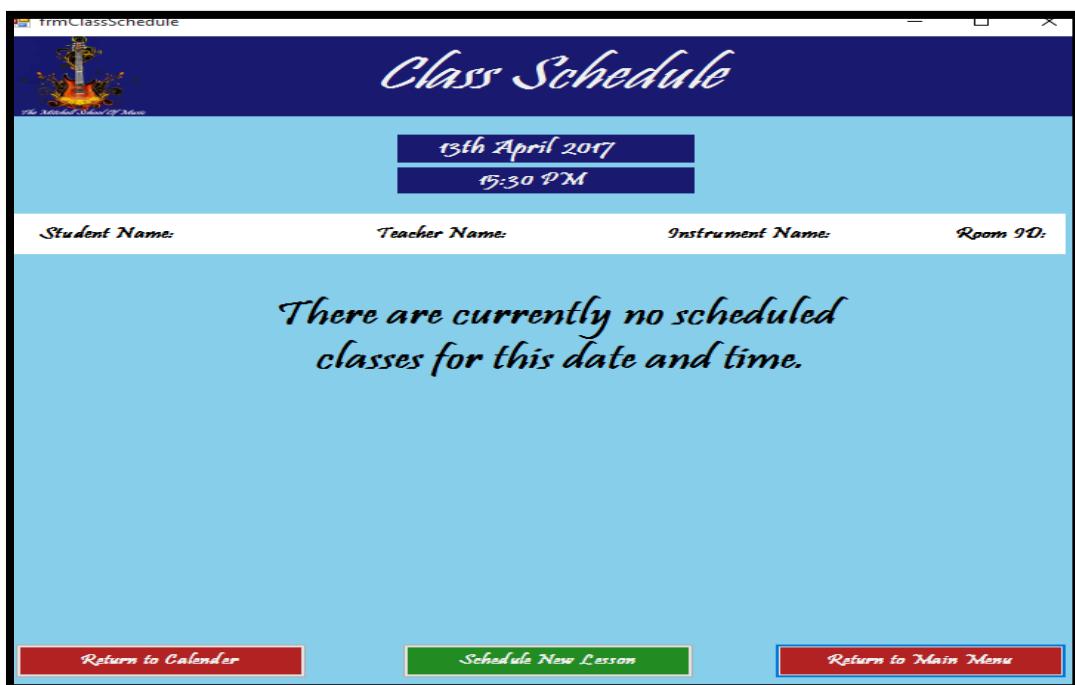
2.15[A][Corrective Action] Screenshot B

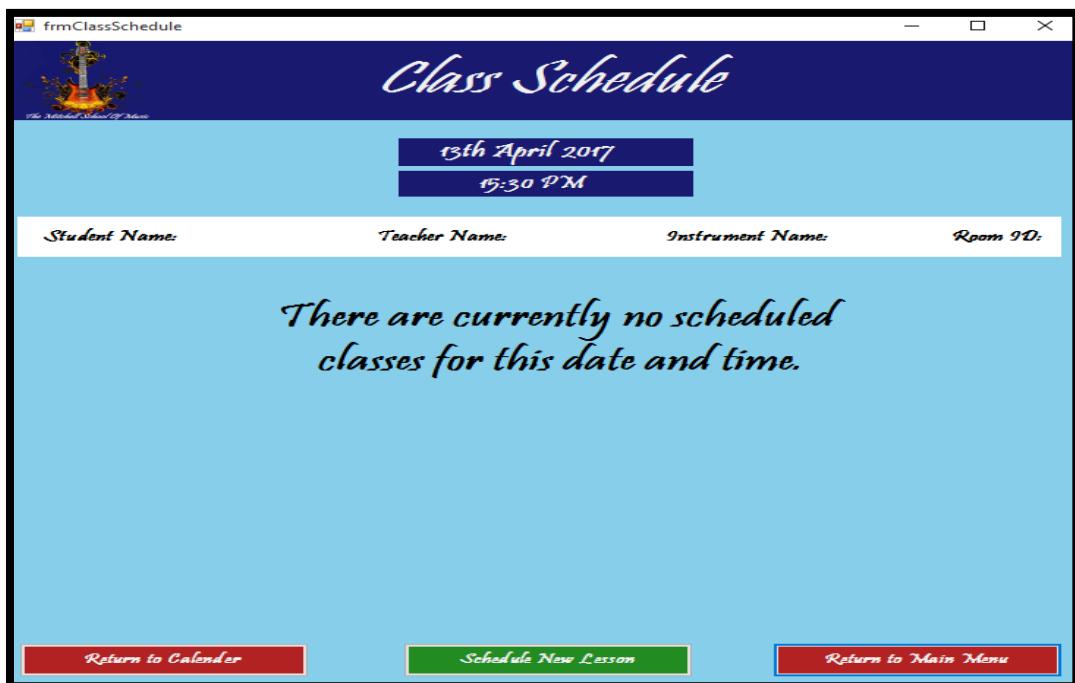
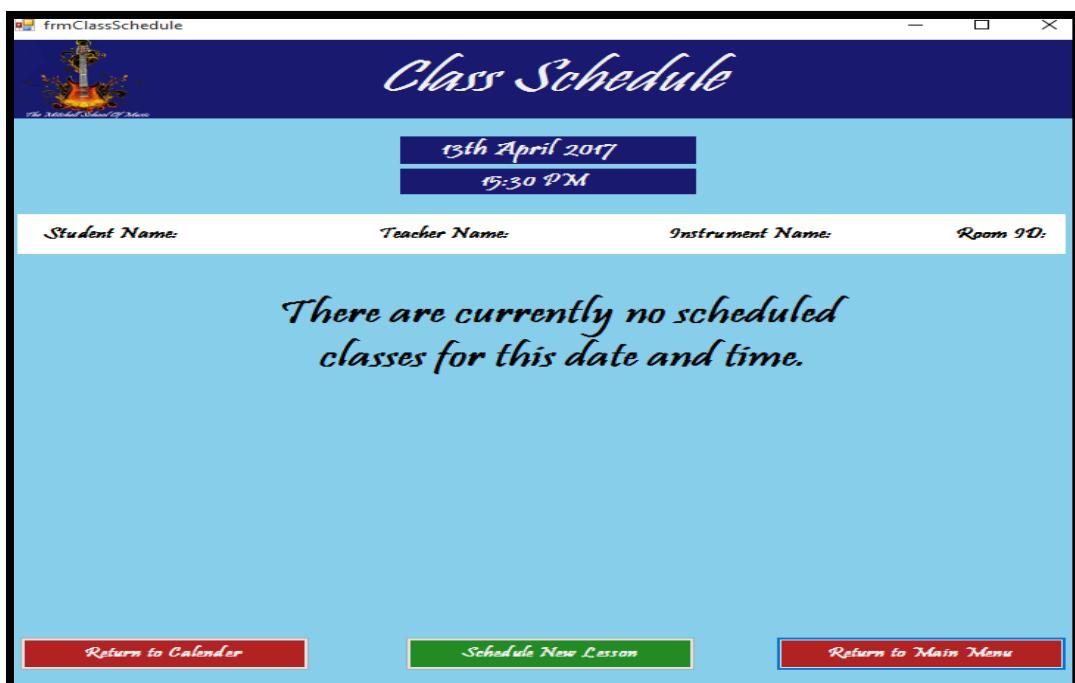
2.15[A][Corrective Action] Screenshot C**Process [C]) Form Navigation****2.15[C][i] – Return To Calendar****2.15[A][i] Screenshot A**

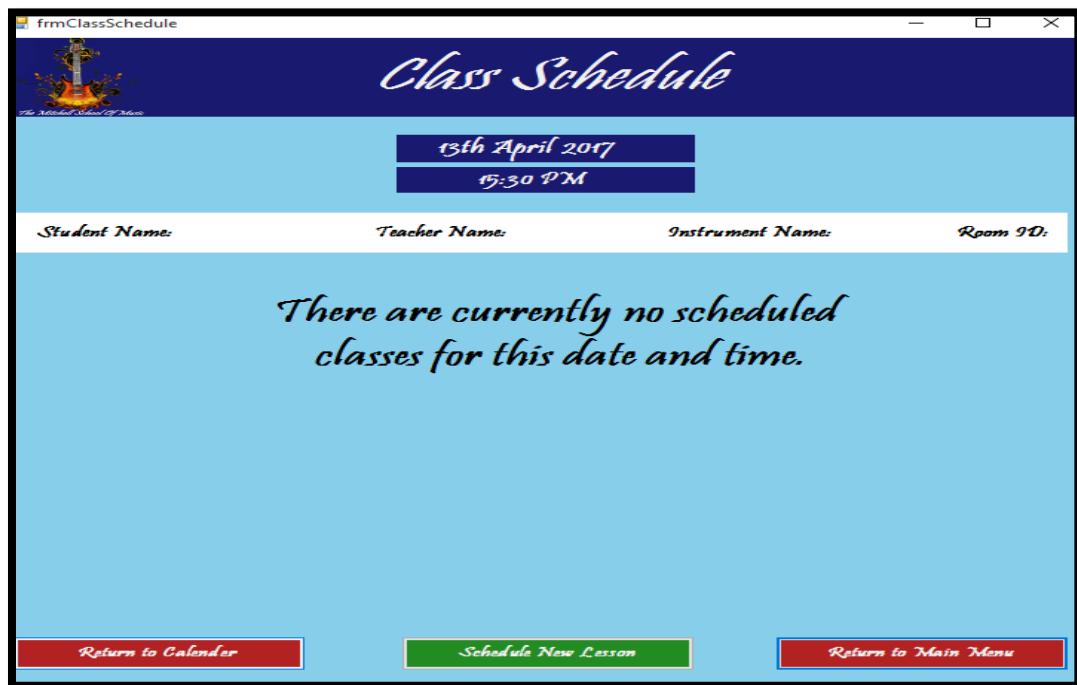
2.15[A][1] Screenshot B**g2.16) Scheduled Calendar – Class Schedule**

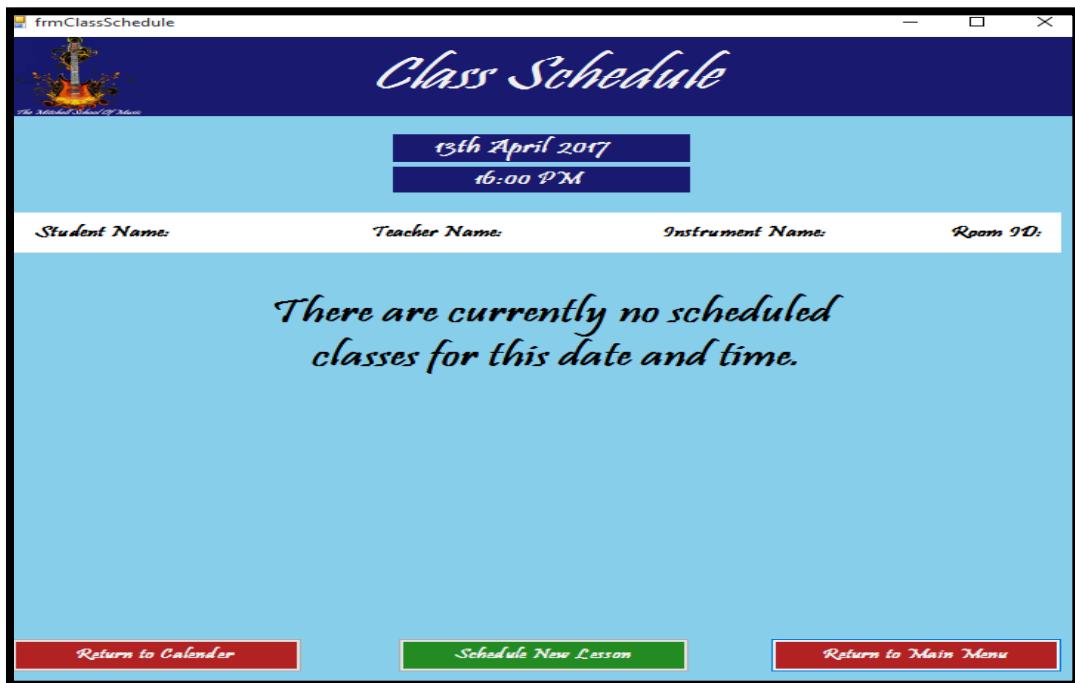
Process [A]) Form Display

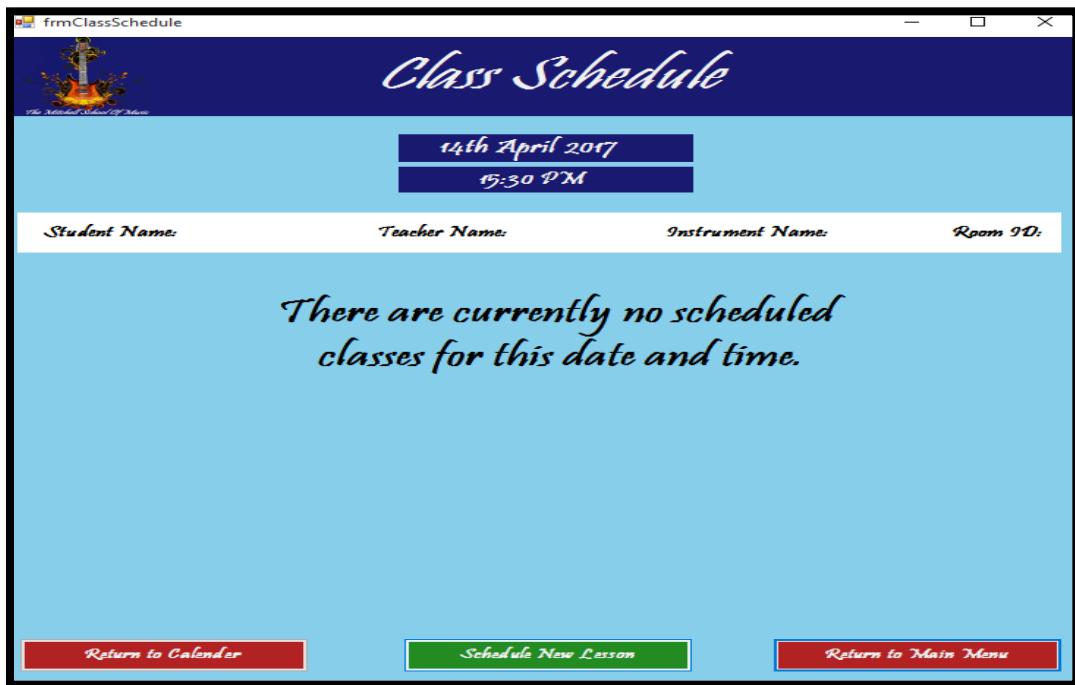
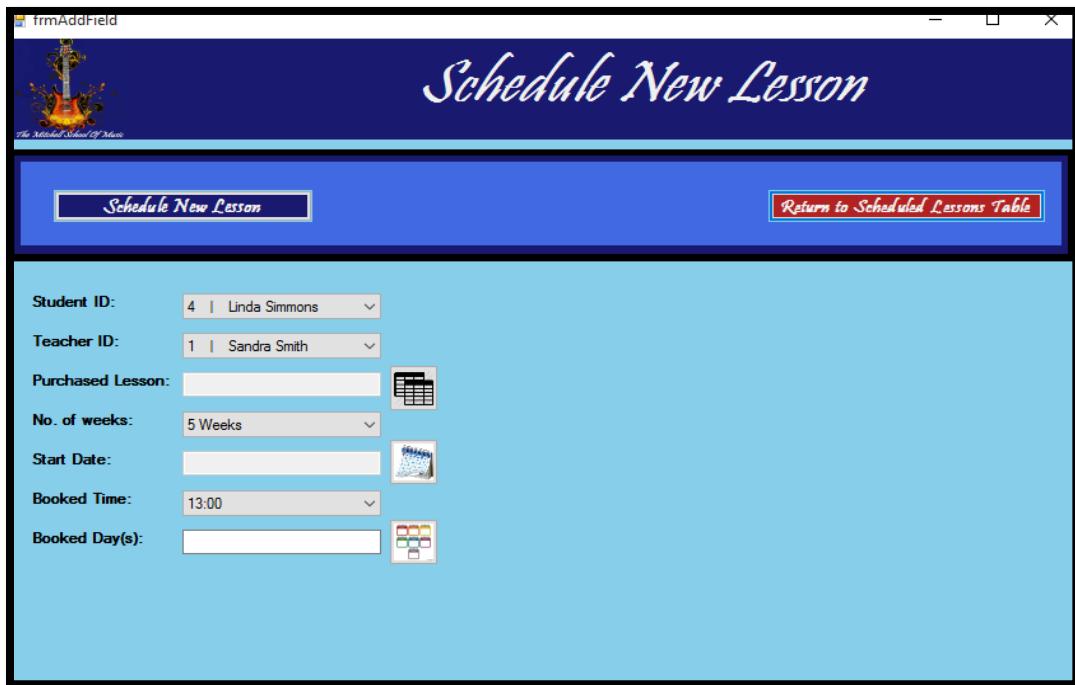
2.16[A][1] – Date Display**2.16[A][1] Screenshot A**

2.16[A][1] Screenshot B**2.16[A][2] – Time Display****2.16[A][2] Screenshot A**

2.16[A][2] Screenshot B**2.16[A][4] – Error Message****2.16[A][4] Screenshot A**

*Process [B]) Form Navigation***2.16[B][1] – Return to Calendar****2.16[A][1] Screenshot A****2.16[A][1] Screenshot B**

2.16[B][2] – Return To Main Menu**2.16[A][2] Screenshot A****2.16[A][2] Screenshot B**

2.16[B][3] – Schedule New Lesson**2.16[A][3] Screenshot A****2.16[A][3] Screenshot B**

2.17) Add New Field Form**2.17[i] Student Record*****Process [A]) Validation******2.17[i][A][1] – First Name Validation******2.17[i][A][1] Screenshot A {Blank}***

The screenshot shows the 'frmAddField' application window titled 'Add Student'. The interface includes a logo for 'The National School Of Music' and two buttons at the top: 'Add New Student' (blue background) and 'Return to Students Table' (red background). The form contains ten input fields with validation messages:

- First Name: Data Required. Please Complete.
- Other Name(s): Data Required. please Complete.
- Surname: Data Required. Please Complete.
- Date Of Birth: Data Required. Please use Calender provided.
- Address: Data Required. Please Complete.
- Town: Data Required. Please Complete.
- PostCode: Data Required. Please Complete.
- Contact Number: Data Required. Please Complete.
- Email Address: The Town value entered is incorrect, please enter text.
- GradeID: Email Must Contain '@'.
- InstrumentID: Tuition Fee Received: (checkbox)

2.17[i][A][1] Screenshot B {Blank}

The screenshot shows the 'frmAddField' application window titled 'Add Student'. The interface includes a logo for 'The National School Of Music' and two buttons at the top: 'Add New Student' (blue background) and 'Return to Students Table' (red background). The form contains ten input fields with validation messages:

- First Name: Test
- Other Name(s): Data Required. please Complete.
- Surname: Data Required. Please Complete.
- Date Of Birth: Data Required. Please use Calender provided.
- Address: Data Required. Please Complete.
- Town: Data Required. Please Complete.
- PostCode: Data Required. Please Complete.
- Contact Number: The Town value entered is incorrect, please enter text.
- Email Address: Email Must Contain '@'.
- GradeID: Tuition Fee Received: (checkbox)
- InstrumentID: Tuition Fee Received: (checkbox)

2.17[i][A][i] Screenshot C [Number]

Add Student

Add New Student **Return to Students Table**

First Name:	Test2	Please Remove Numerical Characters.
Other Name(s):		Data Required. please Complete.
Surname:		Data Required. Please Complete.
Date Of Birth:		Data Required. Please use Calender provided.
Address:		Data Required. Please Complete.
Town:		Data Required. Please Complete.
PostCode:		Data Required. Please Complete.
Contact Number:		The Town value entered is incorrect, please enter text.
Email Address:		Email Must Contain '@'.
GradeID	1	
InstrumentID	1	
Tuition Fee Received	<input type="checkbox"/>	

2.17[i][A][i] Screenshot D [Number]

Add Student

Add New Student **Return to Students Table**

First Name:	Test	
Other Name(s):		Data Required. please Complete.
Surname:		Data Required. Please Complete.
Date Of Birth:		Data Required. Please use Calender provided.
Address:		Data Required. Please Complete.
Town:		Data Required. Please Complete.
PostCode:		Data Required. Please Complete.
Contact Number:		The Town value entered is incorrect, please enter text.
Email Address:		Email Must Contain '@'.
GradeID	1	
InstrumentID	1	
Tuition Fee Received	<input type="checkbox"/>	

2.17[i][A][i] Screenshot E {Symbol}

frmAddField

Add Student

[Add New Student](#) [Return to Students Table](#)

First Name:	Test@	Please Remove Symbols or Punctuation.
Other Name(s):		Data Required, please Complete.
Surname		Data Required, Please Complete.
Date Of Birth:		Data Required, Please use Calender provided.
Address:		Data Required, Please Complete.
Town		Data Required, Please Complete.
PostCode		Data Required, Please Complete.
Contact Number:		The Town value entered is incorrect, please enter text.
Email Address		Email Must Contain '@'.
GradeID	1	
InstrumentID	1	
Tuition Fee Received	<input type="checkbox"/>	

2.17[i][A][i] Screenshot F {Symbol}

frmAddField

Add Student

[Add New Student](#) [Return to Students Table](#)

First Name:	Test	
Other Name(s):		Data Required, please Complete.
Surname		Data Required, Please Complete.
Date Of Birth:		Data Required, Please use Calender provided.
Address:		Data Required, Please Complete.
Town		Data Required, Please Complete.
PostCode		Data Required, Please Complete.
Contact Number:		The Town value entered is incorrect, please enter text.
Email Address		Email Must Contain '@'.
GradeID	1	
InstrumentID	1	
Tuition Fee Received	<input type="checkbox"/>	

2.17[i][A][i] Screenshot G {Punctuation}

The screenshot shows an 'Add Student' form with various fields and their validation messages:

- First Name:** Test" - Validation message: Please Remove Symbols or Punctuation.
- Other Name(s):** - Validation message: Data Required, please Complete.
- Surname:** - Validation message: Data Required, Please Complete.
- Date Of Birth:** - Validation message: Data Required, Please use Calender provided.
- Address:** - Validation message: Data Required, Please Complete.
- Town:** - Validation message: Data Required, Please Complete.
- PostCode:** - Validation message: Data Required, Please Complete.
- Contact Number:** - Validation message: The Town value entered is incorrect, please enter text.
- Email Address:** - Validation message: Email Must Contain '@'.
- GradeID:** 1 - Validation message:
- InstrumentID:** 1 - Validation message:
- Tuition Fee Received:** - Validation message:

2.17[i][A][i] Screenshot H {Punctuation}

The screenshot shows an 'Add Student' form with various fields and their validation messages:

- First Name:** Test - Validation message:
- Other Name(s):** - Validation message: Data Required, please Complete.
- Surname:** - Validation message: Data Required, Please Complete.
- Date Of Birth:** - Validation message: Data Required, Please use Calender provided.
- Address:** - Validation message: Data Required, Please Complete.
- Town:** - Validation message: Data Required, Please Complete.
- PostCode:** - Validation message: Data Required, Please Complete.
- Contact Number:** - Validation message: The Town value entered is incorrect, please enter text.
- Email Address:** - Validation message: Email Must Contain '@'.
- GradeID:** 1 - Validation message:
- InstrumentID:** 1 - Validation message:
- Tuition Fee Received:** - Validation message:

2.17[i][A][1] Screenshot J {Limit}

The screenshot shows the 'Add Student' application window with several validation errors displayed in red text next to the respective input fields:

- First Name:** mit of this textbox to identify the [Textbox]
- Other Name(s):** Data Required, please Complete.
- Surname:** Data Required, Please Complete.
- Date Of Birth:** Data Required, Please use Calendar provided.
- Address:** Data Required, Please Complete.
- Town:** Data Required, Please Complete.
- PostCode:** Data Required, Please Complete.
- Contact Number:** The Town value entered is incorrect, please enter text.
- Email Address:** Email Must Contain '@'.
- GradeID:** 1
- InstrumentID:** 1
- Tuition Fee Received:**

2.17[i][A][1] Screenshot J {Limit}

The screenshot shows the 'Add Student' application window with a 'Word Count' dialog box overlaid. The dialog box displays the following statistics:

Word Count	
Pages	1
Words	9
Characters (no spaces)	41
Characters (with spaces)	49
Paragraphs	1
Lines	1

Include textboxes, footnotes and endnotes

2.17[i][A][1] Screenshot J {Limit}

2.17[i][A][2] – Other Name(s) Validation**2.17[i][A][2] Screenshot A {Blank}**

frmAddField

Add Student

[Add New Student](#) [Return to Students Table](#)

First Name:	<input type="text"/>	Data Required. Please Complete.
Other Name(s):	<input type="text"/>	Data Required. please Complete.
Surname:	<input type="text"/>	Data Required. Please Complete.
Date Of Birth:	<input type="text"/>	Data Required. Please use Calender provided.
Address:	<input type="text"/>	Data Required. Please Complete.
Town:	<input type="text"/>	Data Required. Please Complete.
PostCode:	<input type="text"/>	Data Required. Please Complete.
Contact Number:	<input type="text"/>	The Town value entered is incorrect, please enter text.
Email Address:	<input type="text"/>	Email Must Contain '@'.
GradeID	1	
InstrumentID	1	
Tuition Fee Received <input type="checkbox"/>		

2.17[i][A][2] Screenshot B {Blank}

frmAddField

Add Student

[Add New Student](#) [Return to Students Table](#)

First Name:	<input type="text"/>	Data Required. Please Complete..
Other Name(s):	Test <input type="text"/>	
Surname:	<input type="text"/>	Data Required. Please Complete..
Date Of Birth:	<input type="text"/>	Data Required. Please use Calender provided..
Address:	<input type="text"/>	Data Required. Please Complete..
Town:	<input type="text"/>	Data Required. Please Complete..
PostCode:	<input type="text"/>	Data Required. Please Complete..
Contact Number:	<input type="text"/>	The Town value entered is incorrect, please enter text..
Email Address:	<input type="text"/>	Email Must Contain '@'..
GradeID	1	
InstrumentID	1	
Tuition Fee Received <input type="checkbox"/>		

2.17[i][A][2] Screenshot C {Number}

The screenshot shows the 'Add Student' form with several validation errors:

- First Name:** Data Required. Please Complete.
- Other Name(s):** Please Remove Numerical Characters.
- Surname:** Data Required. Please Complete.
- Date Of Birth:** Data Required. Please use Calender provided.
- Address:** Data Required. Please Complete.
- Town:** Data Required. Please Complete.
- PostCode:** Data Required. Please Complete.
- Contact Number:** The Town value entered is incorrect, please enter text.
- Email Address:** Email Must Contain '@'.
- GradeID:** 1
- Instrument ID:** 1
- Tuition Fee Received:**

2.17[i][A][2] Screenshot D {Number}

The screenshot shows the 'Add Student' form with several validation errors:

- First Name:** Data Required. Please Complete.
- Other Name(s):** Test
- Surname:** Data Required. Please Complete.
- Date Of Birth:** Data Required. Please use Calender provided.
- Address:** Data Required. Please Complete.
- Town:** Data Required. Please Complete.
- PostCode:** Data Required. Please Complete.
- Contact Number:** The Town value entered is incorrect, please enter text.
- Email Address:** Email Must Contain '@'.
- GradeID:** 1
- Instrument ID:** 1
- Tuition Fee Received:**

2.17[i][A][2] Screenshot E {Symbol}

Add Student

Add New Student **Return to Students Table**

First Name:	<input type="text"/>	Data Required. Please Complete.
Other Name(s):	<input type="text"/> Test@	Please remove symbols or punctuation.
Surname:	<input type="text"/>	Data Required. Please Complete.
Date Of Birth:	<input type="text"/>	Data Required. Please use Calender provided.
Address:	<input type="text"/>	Data Required. Please Complete.
Town:	<input type="text"/>	Data Required. Please Complete.
PostCode:	<input type="text"/>	Data Required. Please Complete.
Contact Number:	<input type="text"/>	The Town value entered is incorrect, please enter text.
Email Address:	<input type="text"/>	Email Must Contain '@'.
GradeID:	<input type="text"/> 1	
InstrumentID:	<input type="text"/> 1	
Tuition Fee Received:	<input type="checkbox"/>	

2.17[i][A][2] Screenshot F {Symbol}

Add Student

Add New Student **Return to Students Table**

First Name:	<input type="text"/>	Data Required. Please Complete.
Other Name(s):	<input type="text"/> Test	
Surname:	<input type="text"/>	Data Required. Please Complete.
Date Of Birth:	<input type="text"/>	Data Required. Please use Calender provided.
Address:	<input type="text"/>	Data Required. Please Complete.
Town:	<input type="text"/>	Data Required. Please Complete.
PostCode:	<input type="text"/>	Data Required. Please Complete.
Contact Number:	<input type="text"/>	The Town value entered is incorrect, please enter text.
Email Address:	<input type="text"/>	Email Must Contain '@'.
GradeID:	<input type="text"/> 1	
InstrumentID:	<input type="text"/> 1	
Tuition Fee Received:	<input type="checkbox"/>	

2.17[i][A][2] Screenshot G {Punctuation}

Add Student

Add New Student **Return to Students Table**

First Name:	<input type="text"/>	Data Required. Please Complete..
Other Name(s):	<input type="text"/> Test!"	Please remove symbols or punctuation..
Surname:	<input type="text"/>	Data Required. Please Complete..
Date Of Birth:	<input type="text"/>	Data Required. Please use Calender provided..
Address:	<input type="text"/>	Data Required. Please Complete..
Town:	<input type="text"/>	Data Required. Please Complete..
PostCode:	<input type="text"/>	Data Required. Please Complete..
Contact Number:	<input type="text"/>	The Town value entered is incorrect, please enter text..
Email Address:	<input type="text"/>	Email Must Contain '@'..
GradeID:	<input type="text"/> 1	
InstrumentID:	<input type="text"/> 1	
Tuition Fee Received <input type="checkbox"/>		

2.17[i][A][2] Screenshot H {Punctuation}

Add Student

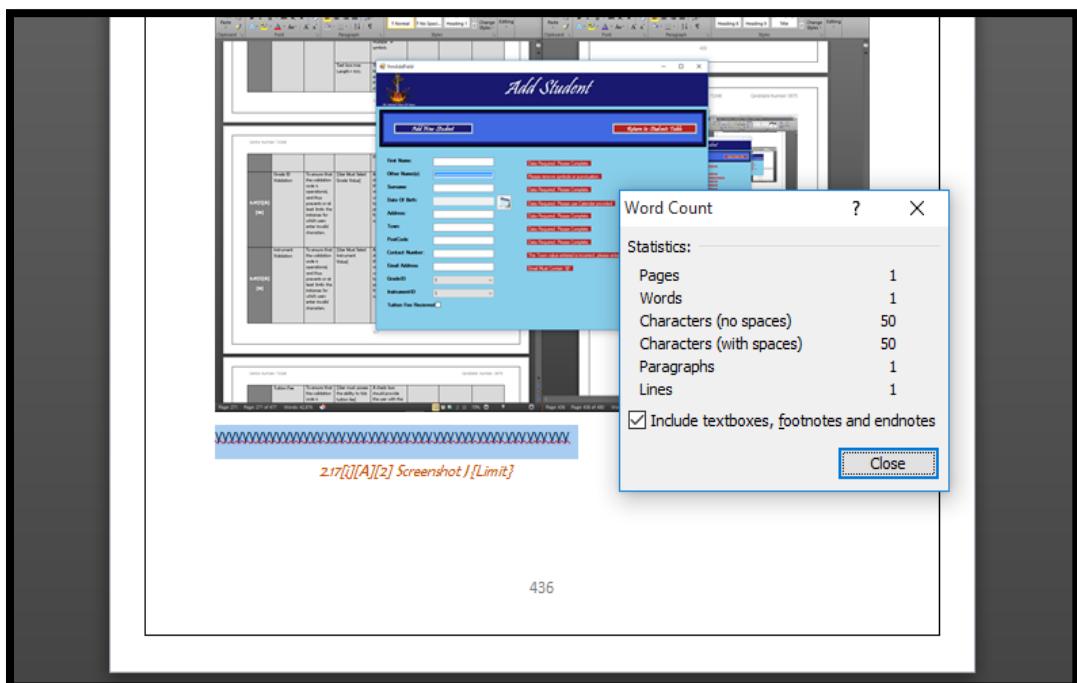
Add New Student **Return to Students Table**

First Name:	<input type="text"/>	Data Required. Please Complete..
Other Name(s):	<input type="text"/> Test	Data Required. Please Complete..
Surname:	<input type="text"/>	Data Required. Please Complete..
Date Of Birth:	<input type="text"/>	Data Required. Please use Calender provided..
Address:	<input type="text"/>	Data Required. Please Complete..
Town:	<input type="text"/>	Data Required. Please Complete..
PostCode:	<input type="text"/>	Data Required. Please Complete..
Contact Number:	<input type="text"/>	The Town value entered is incorrect, please enter text..
Email Address:	<input type="text"/>	Email Must Contain '@'..
GradeID:	<input type="text"/> 1	
InstrumentID:	<input type="text"/> 1	
Tuition Fee Received <input type="checkbox"/>		

2.17[i][A][2] Screenshot I {Limit}

The screenshot shows the 'Add Student' form with various fields and their validation messages:

- First Name:** Data Required. Please Complete.
- Other Name(s):** Please remove symbols or punctuation.
- Surname:** Data Required. Please Complete.
- Date Of Birth:** Data Required. Please use Calender provided.
- Address:** Data Required. Please Complete.
- Town:** Data Required. Please Complete.
- PostCode:** Data Required. Please Complete.
- Contact Number:** The Town value entered is incorrect, please enter text..
- Email Address:** Email Must Contain '@'.
- GradeID:** 1
- InstrumentID:** 1
- Tuition Fee Received:**

2.17[i][A][2] Screenshot J {Limit}

2.17[i][A][3] – Surname Validation**2.17[i][A][3] Screenshot A {Blank}**

frmAddField

Add Student

[Add New Student](#) [Return to Students Table](#)

First Name:	<input type="text"/>	Data Required. Please Complete.
Other Name(s):	<input type="text"/>	Data Required. please Complete.
Surname:	<input type="text"/>	Data Required. Please Complete.
Date Of Birth:	<input type="text"/>	Data Required. Please use Calender provided.
Address:	<input type="text"/>	Data Required. Please Complete.
Town:	<input type="text"/>	Data Required. Please Complete.
PostCode:	<input type="text"/>	Data Required. Please Complete.
Contact Number:	<input type="text"/>	The Town value entered is incorrect, please enter text.
Email Address:	<input type="text"/>	Email Must Contain '@'.
GradeID	1 <input type="button" value="▼"/>	
InstrumentID	1 <input type="button" value="▼"/>	
Tuition Fee Received	<input type="checkbox"/>	

2.17[i][A][3] Screenshot B {Blank}

frmAddField

Add Student

[Add New Student](#) [Return to Students Table](#)

First Name:	<input type="text"/>	Data Required. Please Complete.
Other Name(s):	<input type="text"/>	Data Required. please Complete.
Surname:	Test <input type="text"/>	
Date Of Birth:	<input type="text"/>	Data Required. Please use Calender provided.
Address:	<input type="text"/>	Data Required. Please Complete.
Town:	<input type="text"/>	Data Required. Please Complete.
PostCode:	<input type="text"/>	Data Required. Please Complete.
Contact Number:	<input type="text"/>	The Town value entered is incorrect, please enter text.
Email Address:	<input type="text"/>	Email Must Contain '@'.
GradeID	1 <input type="button" value="▼"/>	
InstrumentID	1 <input type="button" value="▼"/>	
Tuition Fee Received	<input type="checkbox"/>	

2.17[i][A][3] Screenshot C {Number}

frmAddField

Add Student

[Add New Student](#) [Return to Students Table](#)

First Name:	<input type="text"/>	Data Required. Please Complete.
Other Name(s):	<input type="text"/>	Data Required. please Complete.
Surname:	<input type="text"/> Test2	Please Remove Numerical Characters..
Date Of Birth:	<input type="text"/>	Data Required. Please use Calender provided.
Address:	<input type="text"/>	Data Required. Please Complete.
Town:	<input type="text"/>	Data Required. Please Complete.
PostCode:	<input type="text"/>	Data Required. Please Complete.
Contact Number:	<input type="text"/>	The Town value entered is incorrect, please enter text..
Email Address:	<input type="text"/>	Email Must Contain '@'.
GradeID:	<input type="text"/> 1	
InstrumentID:	<input type="text"/> 1	
Tuition Fee Received:	<input type="checkbox"/>	

2.17[i][A][3] Screenshot D {Number}

frmAddField

Add Student

[Add New Student](#) [Return to Students Table](#)

First Name:	<input type="text"/>	Data Required. Please Complete.
Other Name(s):	<input type="text"/>	Data Required. please Complete.
Surname:	<input type="text"/> Test	
Date Of Birth:	<input type="text"/>	Data Required. Please use Calender provided.
Address:	<input type="text"/>	Data Required. Please Complete.
Town:	<input type="text"/>	Data Required. Please Complete.
PostCode:	<input type="text"/>	Data Required. Please Complete.
Contact Number:	<input type="text"/>	The Town value entered is incorrect, please enter text..
Email Address:	<input type="text"/>	Email Must Contain '@'.
GradeID:	<input type="text"/> 1	
InstrumentID:	<input type="text"/> 1	
Tuition Fee Received:	<input type="checkbox"/>	

2.17[i][A][3] Screenshot E {Symbol}

Add Student

Add New Student **Return to Students Table**

First Name:	<input type="text"/>	Data Required. Please Complete.
Other Name(s):	<input type="text"/>	Data Required. please Complete.
Surname:	<input type="text"/> Test@	Please Remove Numerical Characters..
Date Of Birth:	<input type="text"/>	Data Required. Please use Calender provided.
Address:	<input type="text"/>	Data Required. Please Complete.
Town:	<input type="text"/>	Data Required. Please Complete.
PostCode:	<input type="text"/>	Data Required. Please Complete.
Contact Number:	<input type="text"/>	The Town value entered is incorrect, please enter text..
Email Address:	<input type="text"/>	Email Must Contain '@'.
GradeID:	<input type="text"/> 1	
InstrumentID:	<input type="text"/> 1	
Tuition Fee Received:	<input type="checkbox"/>	

2.17[i][A][3] Screenshot F {Symbol}

Add Student

Add New Student **Return to Students Table**

First Name:	<input type="text"/>	Data Required. Please Complete.
Other Name(s):	<input type="text"/>	Data Required. please Complete.
Surname:	<input type="text"/> Test	
Date Of Birth:	<input type="text"/>	Data Required. Please use Calender provided.
Address:	<input type="text"/>	Data Required. Please Complete.
Town:	<input type="text"/>	Data Required. Please Complete.
PostCode:	<input type="text"/>	Data Required. Please Complete.
Contact Number:	<input type="text"/>	The Town value entered is incorrect, please enter text..
Email Address:	<input type="text"/>	Email Must Contain '@'.
GradeID:	<input type="text"/> 1	
InstrumentID:	<input type="text"/> 1	
Tuition Fee Received:	<input type="checkbox"/>	

2.17[i][A][3] Screenshot G {Punctuation}

The screenshot shows an 'Add Student' form with various input fields and validation messages. The validation messages include:

- First Name: Data Required. Please Complete.
- Other Name(s): Data Required. please Complete.
- Surname: please Remove Symbols or punctuation.
- Date Of Birth: Data Required. Please use Calender provided.
- Address: Data Required. Please Complete.
- Town: Data Required. Please Complete.
- PostCode: Data Required. Please Complete.
- Contact Number: The Town value entered is incorrect, please enter text..
- Email Address: Email Must Contain '@'.
- GradeID: 1
- InstrumentID: 1
- Tuition Fee Received:

2.17[i][A][3] Screenshot H {Punctuation}

The screenshot shows the same 'Add Student' form as in screenshot G, but with corrected punctuation in the Surname field. The validation message for Surname now reads: "please Remove Symbols or punctuation." The other fields and their validation messages remain the same as in screenshot G.

2.17[i][A][3] Screenshot J {Limit}

Add Student

Add New Student **Return to Students Table**

First Name: Data Required. Please Complete.

Other Name(s): Data Required. Please Complete.

Surname: please Remove Symbols or punctuation.

Date Of Birth: Data Required. Please use Calender provided.

Address: Data Required. Please Complete.

Town: Data Required. Please Complete.

PostCode: Data Required. Please Complete.

Contact Number: The Town value entered is incorrect, please enter text..

Email Address: Email Must Contain '@'.

GradeID:

InstrumentID:

Tuition Fee Received:

2.17[i][A][3] Screenshot J {Limit}

Word Count

Statistics:

Pages	1
Words	1
Characters (no spaces)	50
Characters (with spaces)	50
Paragraphs	1
Lines	1

Include textboxes, footnotes and endnotes

Close

441

ICT, Unit 8 Centre Number: 71348 Candidate Number: 0075

2.17[i][A][4] – Date Of Birth Validation

2.17[i][A][4] – Date Of Birth Validation**2.17[i][A][4] Screenshot A {Blank}**

Add Student

Add New Student **Return to Students Table**

First Name:	<input type="text"/>	Data Required. Please Complete.
Other Name(s):	<input type="text"/>	Data Required. please Complete.
Surname:	<input type="text"/>	Data Required. Please Complete.
Date Of Birth:	<input type="text"/>	Data Required. Please use Calender provided.
Address:	<input type="text"/>	Data Required. Please Complete.
Town:	<input type="text"/>	Data Required. Please Complete.
PostCode:	<input type="text"/>	Data Required. Please Complete.
Contact Number:	<input type="text"/>	The Town value entered is incorrect, please enter text.
Email Address:	<input type="text"/>	Email Must Contain '@'.
GradeID:	1 <input type="button" value="▼"/>	
InstrumentID:	1 <input type="button" value="▼"/>	
Tuition Fee Received:	<input type="checkbox"/>	

2.17[i][A][4] Screenshot B {Blank}

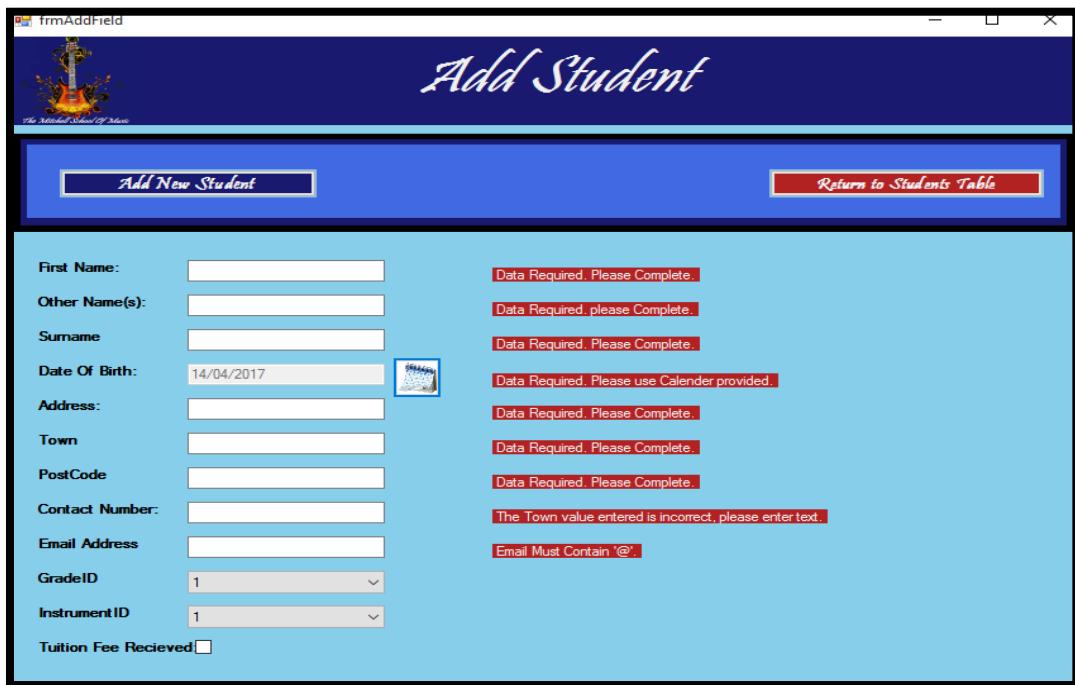
Add Student

Add New Student **Return to Students Table**

First Name:	<input type="text"/>	
Other Name(s):	<input type="text"/>	
Surname:	<input type="text"/>	
Date Of Birth:	<input type="text"/>	
Address:	<input type="text"/>	
Town:	<input type="text"/>	Data Required. Please Complete.
PostCode:	<input type="text"/>	Data Required. Please Complete.
Contact Number:	<input type="text"/>	The Town value entered is incorrect, please enter text.
Email Address:	<input type="text"/>	Email Must Contain '@'.
GradeID:	1 <input type="button" value="▼"/>	
InstrumentID:	1 <input type="button" value="▼"/>	
Tuition Fee Received:	<input type="checkbox"/>	

April 2017						
Mon	Tue	Wed	Thu	Fri	Sat	Sun
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
1	2	3	4	5	6	7

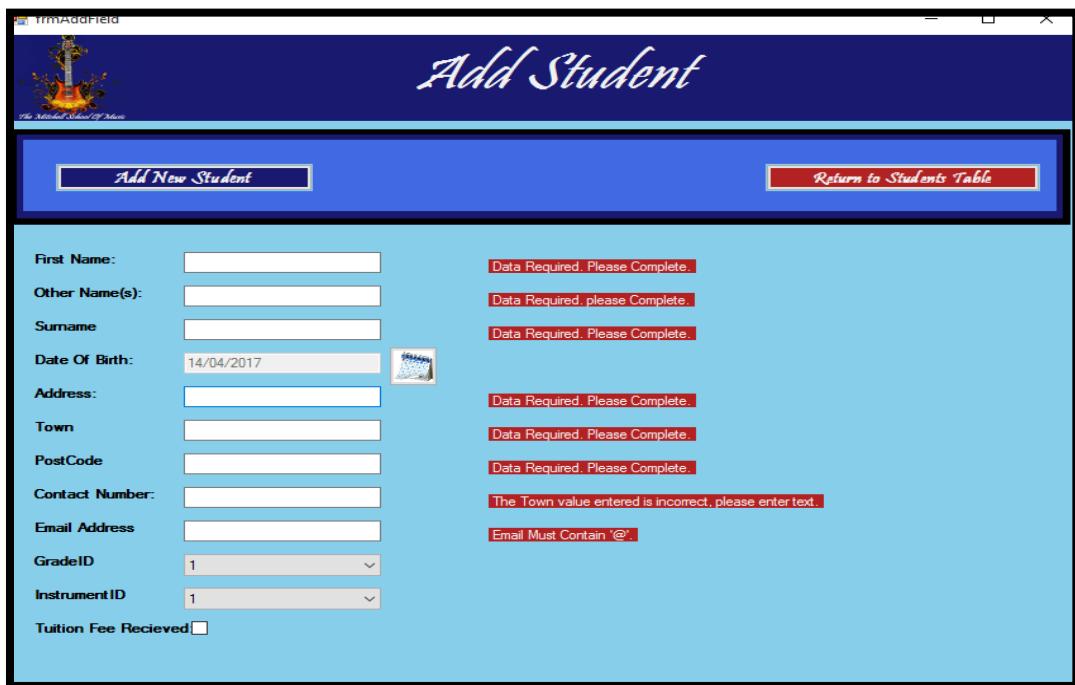
Today: 25/04/2017

2.17[i][A][4] Screenshot C {Blank}


Add Student

Add New Student **Return to Students Table**

First Name:	<input type="text"/>	Data Required. Please Complete..
Other Name(s):	<input type="text"/>	Data Required. please Complete..
Surname:	<input type="text"/>	Data Required. Please Complete..
Date Of Birth:	<input type="text"/> 14/04/2017	Data Required. Please use Calender provided..
Address:	<input type="text"/>	Data Required. Please Complete..
Town:	<input type="text"/>	Data Required. Please Complete..
PostCode:	<input type="text"/>	Data Required. Please Complete..
Contact Number:	<input type="text"/>	The Town value entered is incorrect, please enter text..
Email Address:	<input type="text"/>	Email Must Contain '@'..
GradeID:	<input type="text"/> 1	
InstrumentID:	<input type="text"/> 1	
Tuition Fee Received:	<input type="checkbox"/>	

2.17[i][A][5] – Address Validation**2.17[i][A][5] Screenshot A {Blank}**


Add Student

Add New Student **Return to Students Table**

First Name:	<input type="text"/>	Data Required. Please Complete..
Other Name(s):	<input type="text"/>	Data Required. please Complete..
Surname:	<input type="text"/>	Data Required. Please Complete..
Date Of Birth:	<input type="text"/> 14/04/2017	
Address:	<input type="text"/>	Data Required. Please Complete..
Town:	<input type="text"/>	Data Required. Please Complete..
PostCode:	<input type="text"/>	Data Required. Please Complete..
Contact Number:	<input type="text"/>	The Town value entered is incorrect, please enter text..
Email Address:	<input type="text"/>	Email Must Contain '@'..
GradeID:	<input type="text"/> 1	
InstrumentID:	<input type="text"/> 1	
Tuition Fee Received:	<input type="checkbox"/>	

2.17[i][A][5] Screenshot B {Blank}

Add Student

Add New Student **Return to Students Table**

First Name:	<input type="text"/>	Data Required. Please Complete.
Other Name(s):	<input type="text"/>	Data Required. please Complete.
Surname	<input type="text"/>	Data Required. Please Complete.
Date Of Birth:	<input type="text" value="14/04/2017"/>	
Address:	<input type="text" value="123 Test"/>	
Town	<input type="text"/>	Data Required. Please Complete.
PostCode	<input type="text"/>	Data Required. Please Complete.
Contact Number:	<input type="text"/>	The Town value entered is incorrect, please enter text.
Email Address	<input type="text"/>	Email Must Contain '@'.
GradeID	<input type="text" value="1"/>	
InstrumentID	<input type="text" value="1"/>	
Tuition Fee Received	<input type="checkbox"/>	

2.17[i][A][5] Screenshot C {Numbers - Letters}

Add Student

Add New Student **Return to Students Table**

First Name:	<input type="text"/>	Data Required. Please Complete.
Other Name(s):	<input type="text"/>	Data Required. please Complete.
Surname	<input type="text"/>	Data Required. Please Complete.
Date Of Birth:	<input type="text" value="14/04/2017"/>	
Address:	<input type="text" value="12y Test"/>	Please Remove alphabetical Characters from street number.
Town	<input type="text"/>	Data Required. Please Complete.
PostCode	<input type="text"/>	Data Required. Please Complete.
Contact Number:	<input type="text"/>	The Town value entered is incorrect, please enter text.
Email Address	<input type="text"/>	Email Must Contain '@'.
GradeID	<input type="text" value="1"/>	
InstrumentID	<input type="text" value="1"/>	
Tuition Fee Received	<input type="checkbox"/>	

2.17[i][A][5] Screenshot D { Numbers - Letters }

Add Student

Add New Student **Return to Students Table**

First Name:	<input type="text"/>	Data Required. Please Complete.
Other Name(s):	<input type="text"/>	Data Required. please Complete.
Surname:	<input type="text"/>	Data Required. Please Complete.
Date Of Birth:	14/04/2017	
Address:	123 Test	
Town:	<input type="text"/>	Data Required. Please Complete.
PostCode:	<input type="text"/>	Data Required. Please Complete.
Contact Number:	<input type="text"/>	The Town value entered is incorrect, please enter text..
Email Address:	<input type="text"/>	Email Must Contain '@'..
GradeID:	1	
InstrumentID:	1	
Tuition Fee Received:	<input type="checkbox"/>	

2.17[i][A][5] Screenshot E {Letters - Numbers}

Add Student

Add New Student **Return to Students Table**

First Name:	<input type="text"/>	Data Required. Please Complete.
Other Name(s):	<input type="text"/>	Data Required. please Complete.
Surname:	<input type="text"/>	Data Required. Please Complete.
Date Of Birth:	14/04/2017	
Address:	124 Test2	Please Remove Numerical Characters from street name..
Town:	<input type="text"/>	Data Required. Please Complete.
PostCode:	<input type="text"/>	Data Required. Please Complete.
Contact Number:	<input type="text"/>	The Town value entered is incorrect, please enter text..
Email Address:	<input type="text"/>	Email Must Contain '@'..
GradeID:	1	
InstrumentID:	1	
Tuition Fee Received:	<input type="checkbox"/>	

2.17[i][A][5] Screenshot F {Letters - Number}

Add Student

Add New Student **Return to Students Table**

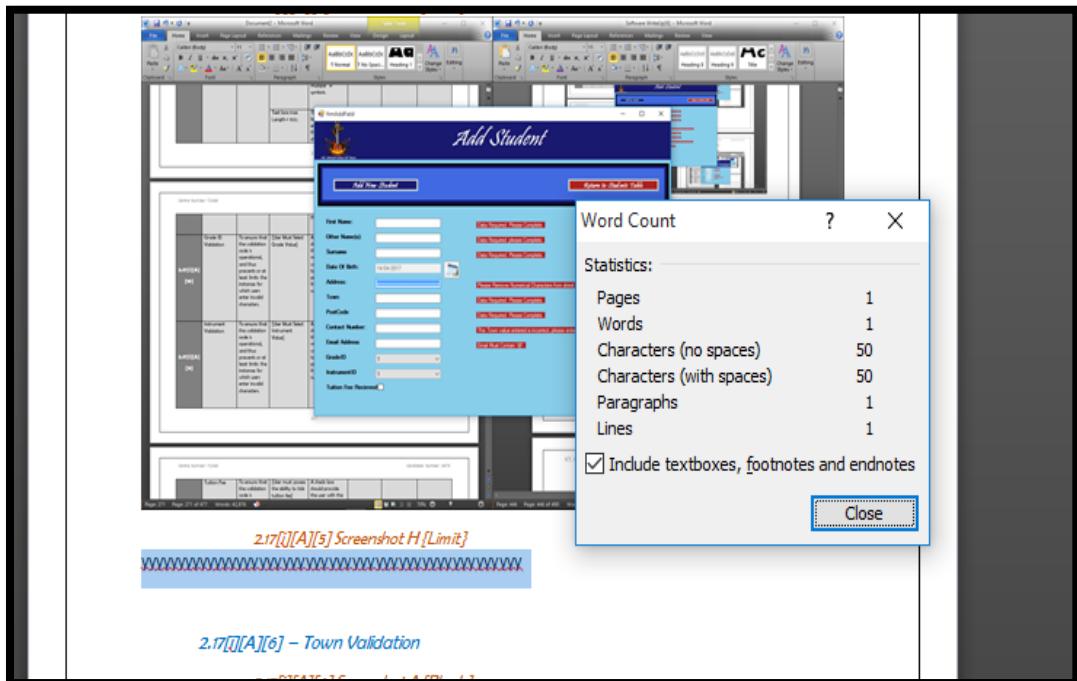
First Name:	<input type="text"/>	Data Required. Please Complete.
Other Name(s):	<input type="text"/>	Data Required. please Complete.
Surname:	<input type="text"/>	Data Required. Please Complete.
Date Of Birth:	14/04/2017	
Address:	123 Test	
Town:	<input type="text"/>	Data Required. Please Complete.
PostCode:	<input type="text"/>	Data Required. Please Complete.
Contact Number:	<input type="text"/>	The Town value entered is incorrect, please enter text.
Email Address:	<input type="text"/>	Email Must Contain '@'.
GradeID:	1	
InstrumentID:	1	
Tuition Fee Received:	<input type="checkbox"/>	

2.17[i][A][5] Screenshot G {Limit}

Add Student

Add New Student **Return to Students Table**

First Name:	<input type="text"/>	Data Required. Please Complete.
Other Name(s):	<input type="text"/>	Data Required. please Complete.
Surname:	<input type="text"/>	Data Required. Please Complete.
Date Of Birth:	14/04/2017	
Address:	oooooooooooooooooooo	Please Remove Numerical Characters from street name.
Town:	<input type="text"/>	Data Required. Please Complete.
PostCode:	<input type="text"/>	Data Required. Please Complete.
Contact Number:	<input type="text"/>	The Town value entered is incorrect, please enter text.
Email Address:	<input type="text"/>	Email Must Contain '@'.
GradeID:	1	
InstrumentID:	1	
Tuition Fee Received:	<input type="checkbox"/>	

2.17[i][A][5] Screenshot H {Limit}**2.17[i][A][6] – Town Validation****2.17[i][A][6] – Town Validation****2.17[i][A][6] Screenshot A {Blank}**

Add Student

Add New Student **Return to Students Table**

First Name:	<input type="text"/>	Data Required. Please Complete.
Other Name(s):	<input type="text"/>	Data Required. Please Complete.
Surname:	<input type="text"/>	Data Required. Please Complete.
Date Of Birth:	14/04/2017	
Address:	<input type="text"/>	Data Required. Please Complete.
Town:	<input type="text"/>	Data Required. Please Complete.
PostCode:	<input type="text"/>	Data Required. Please Complete.
Contact Number:	<input type="text"/>	The Town value entered is incorrect, please enter text..
Email Address:	<input type="text"/>	Email Must Contain '@'..
GradeID	1	
InstrumentID	1	
Tuition Fee Received	<input type="checkbox"/>	

2.17[i][A][6] Screenshot B {Blank}

Add Student

Add New Student **Return to Students Table**

First Name:	<input type="text"/>	Data Required. Please Complete.
Other Name(s):	<input type="text"/>	Data Required. please Complete.
Surname	<input type="text"/>	Data Required. Please Complete.
Date Of Birth:	<input type="text"/> 14/04/2017	
Address:	<input type="text"/>	Data Required. Please Complete.
Town	<input type="text"/> Test	
PostCode	<input type="text"/>	Data Required. Please Complete.
Contact Number:	<input type="text"/>	
Email Address	<input type="text"/>	Email Must Contain '@'.
GradeID	<input type="text"/> 1	
InstrumentID	<input type="text"/> 1	
Tuition Fee Received	<input type="checkbox"/>	

2.17[i][A][6] Screenshot C {Number}

Add Student

Add New Student **Return to Students Table**

First Name:	<input type="text"/>	Data Required. Please Complete.
Other Name(s):	<input type="text"/>	Data Required. please Complete.
Surname	<input type="text"/>	Data Required. Please Complete.
Date Of Birth:	<input type="text"/> 14/04/2017	
Address:	<input type="text"/>	Data Required. Please Complete.
Town	<input type="text"/> Test2	Please remove numerical characters.
PostCode	<input type="text"/>	Data Required. Please Complete.
Contact Number:	<input type="text"/>	
Email Address	<input type="text"/>	Email Must Contain '@'.
GradeID	<input type="text"/> 1	
InstrumentID	<input type="text"/> 1	
Tuition Fee Received	<input type="checkbox"/>	

2.17[i][A][6] Screenshot D {Number}

Add Student

Add New Student **Return to Students Table**

First Name:	<input type="text"/>	Data Required. Please Complete.
Other Name(s):	<input type="text"/>	Data Required. please Complete.
Surname:	<input type="text"/>	Data Required. Please Complete..
Date Of Birth:	14/04/2017	
Address:	<input type="text"/>	Data Required. Please Complete..
Town:	Test	
PostCode:	<input type="text"/>	Data Required. Please Complete..
Contact Number:	<input type="text"/>	
Email Address:	<input type="text"/>	Email Must Contain '@'.
GradeID:	1	
InstrumentID:	1	
Tuition Fee Received:	<input type="checkbox"/>	

2.17[i][A][6] Screenshot E {Symbol}

Add Student

Add New Student **Return to Students Table**

First Name:	<input type="text"/>	Data Required. Please Complete..
Other Name(s):	<input type="text"/>	Data Required. please Complete..
Surname:	<input type="text"/>	Data Required. Please Complete..
Date Of Birth:	14/04/2017	
Address:	<input type="text"/>	Data Required. Please Complete..
Town:	Test@	Please remove symbols and punctuation.
PostCode:	<input type="text"/>	Data Required. Please Complete..
Contact Number:	<input type="text"/>	
Email Address:	<input type="text"/>	Email Must Contain '@'.
GradeID:	1	
InstrumentID:	1	
Tuition Fee Received:	<input type="checkbox"/>	

2.17[i][A][6] Screenshot F {Symbol}

frmAddField Add Student

Add New Student Return to Students Table

First Name:	<input type="text"/>	Data Required. Please Complete..
Other Name(s):	<input type="text"/>	Data Required. please Complete..
Surname:	<input type="text"/>	Data Required. Please Complete..
Date Of Birth:	<input type="text"/> 14/04/2017	
Address:	<input type="text"/>	Data Required. Please Complete..
Town:	<input type="text"/> Test	
PostCode:	<input type="text"/>	Data Required. Please Complete..
Contact Number:	<input type="text"/>	
Email Address:	<input type="text"/>	Email Must Contain '@'..
GradeID:	<input type="text"/> 1	
InstrumentID:	<input type="text"/> 1	
Tuition Fee Received:	<input checked="" type="checkbox"/>	

2.17[i][A][6] Screenshot G {Punctuation}

frmAddField Add Student

Add New Student Return to Students Table

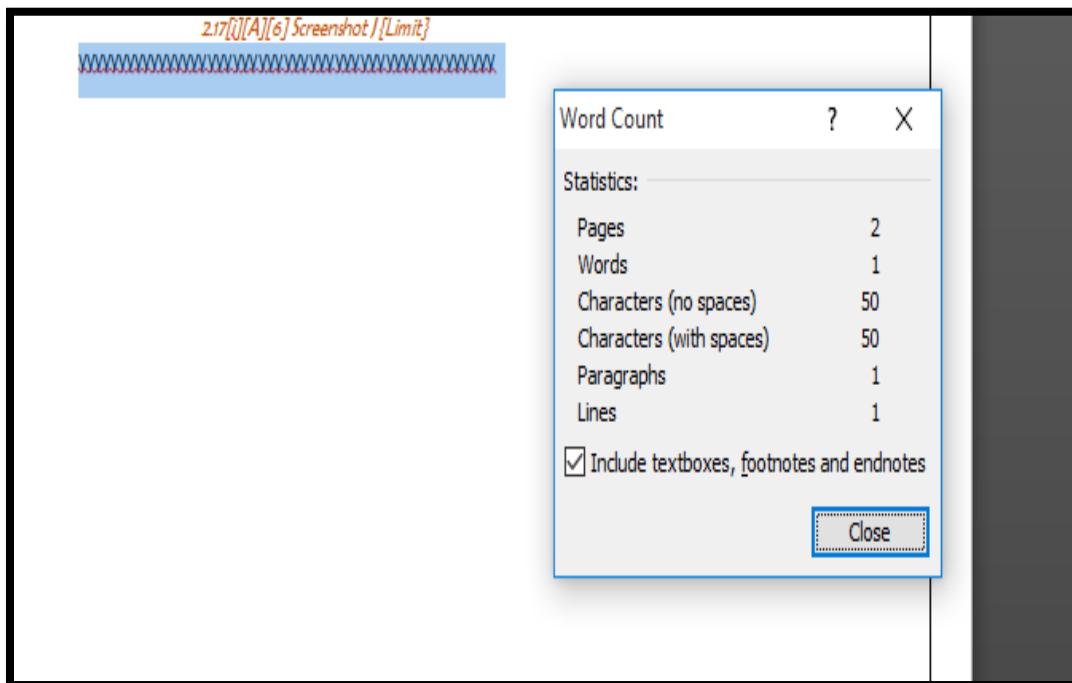
First Name:	<input type="text"/>	Data Required. Please Complete..
Other Name(s):	<input type="text"/>	Data Required. please Complete..
Surname:	<input type="text"/>	Data Required. Please Complete..
Date Of Birth:	<input type="text"/> 14/04/2017	
Address:	<input type="text"/>	Data Required. Please Complete..
Town:	<input type="text"/> Test"!	Please remove symbols and punctuation..
PostCode:	<input type="text"/>	Data Required. Please Complete..
Contact Number:	<input type="text"/>	
Email Address:	<input type="text"/>	Email Must Contain '@'..
GradeID:	<input type="text"/> 1	
InstrumentID:	<input type="text"/> 1	
Tuition Fee Received:	<input checked="" type="checkbox"/>	

2.17[i][A][6] Screenshot H {Punctuation}

The screenshot shows an 'Add Student' form with various fields and validation messages. The fields include First Name, Other Name(s), Surname, Date Of Birth, Address, Town, PostCode, Contact Number, Email Address, GradeID, InstrumentID, and Tuition Fee Received. Red validation messages are displayed next to each field: 'Data Required. Please Complete.' for First Name, Other Name(s), and Surname; 'Data Required. please Complete.' for Date Of Birth; 'Data Required. Please Complete.' for Address; 'Data Required. Please Complete.' for PostCode; 'Email Must Contain '@''. The form has a blue header and a red 'Return to Students Table' button.

2.17[i][A][6] Screenshot I {Limit}

The screenshot shows the same 'Add Student' form as in Screenshot H, but with different validation messages. The validation messages now include 'Data Required. Please Complete.' for First Name, Other Name(s), and Surname; 'Please remove symbols and punctuation.' for Town (which contains a wavy underline); and 'Data Required. Please Complete.' for Contact Number. The rest of the validation messages remain the same as in Screenshot H.

2.17[i][A][6] Screenshot J {Limit}**2.17[i][A][7] – Post Code Validation****2.17[i][A][7] Screenshot A {Blank}**

A screenshot of a Windows application window titled 'frmAddField'. The title bar features a logo of a guitar and the text 'The Musical School Of Wales'. The main title of the window is 'Add Student'. The window contains an 'Add New Student' button in the top left and a 'Return to Students Table' button in the top right. The form itself has several text input fields and dropdown menus, each with a red validation message to its right. The fields and their messages are:

- First Name: Data Required. Please Complete.
- Other Name(s): Data Required. please Complete.
- Surname: Data Required. Please Complete.
- Date Of Birth: 14/04/2017 (next to a small calendar icon)
- Address: Data Required. Please Complete.
- Town: Data Required. Please Complete.
- PostCode: Data Required. Please Complete.
- Contact Number: The Town value entered is incorrect, please enter text.
- Email Address: Email Must Contain '@'.
- GradeID: 1 (dropdown menu)
- InstrumentID: 1 (dropdown menu)
- Tuition Fee Received:

2.17[i][A][7] Screenshot B {Blank}

Add Student

Add New Student **Return to Students Table**

First Name:	<input type="text"/>	Data Required. Please Complete.
Other Name(s):	<input type="text"/>	Data Required. please Complete.
Surname:	<input type="text"/>	Data Required. Please Complete.
Date Of Birth:	14/04/2017	
Address:	<input type="text"/>	Data Required. Please Complete.
Town:	<input type="text"/>	Data Required. Please Complete.
PostCode:	BT522EP	
Contact Number:	<input type="text"/>	The Town value entered is incorrect, please enter text.
Email Address:	<input type="text"/>	Email Must Contain '@'.
GradeID:	1	
InstrumentID:	1	
Tuition Fee Received:	<input type="checkbox"/>	

2.17[i][A][7] Screenshot C {BT}

Add Student

Add New Student **Return to Students Table**

First Name:	<input type="text"/>	Data Required. Please Complete.
Other Name(s):	<input type="text"/>	Data Required. please Complete.
Surname:	<input type="text"/>	Data Required. Please Complete.
Date Of Birth:	14/04/2017	
Address:	<input type="text"/>	Data Required. Please Complete.
Town:	<input type="text"/>	Data Required. Please Complete.
PostCode:	TT522EP	[PostCode Format = LL000LL] First characters must be 'BT'.
Contact Number:	<input type="text"/>	The Town value entered is incorrect, please enter text.
Email Address:	<input type="text"/>	Email Must Contain '@'.
GradeID:	1	
InstrumentID:	1	
Tuition Fee Received:	<input type="checkbox"/>	

2.17[i][A][7] Screenshot D {BT}

The National School Of Music

Add Student

[Add New Student](#) [Return to Students Table](#)

First Name:	<input type="text"/>	Data Required. Please Complete.
Other Name(s):	<input type="text"/>	Data Required. please Complete.
Surname	<input type="text"/>	Data Required. Please Complete.
Date Of Birth:	14/04/2017	
Address:	<input type="text"/>	Data Required. Please Complete.
Town	<input type="text"/>	Data Required. Please Complete.
PostCode	BT522EP	
Contact Number:	<input type="text"/>	The Town value entered is incorrect, please enter text.
Email Address	<input type="text"/>	Email Must Contain '@'.
GradeID	1	
InstrumentID	1	
Tuition Fee Received	<input checked="" type="checkbox"/>	

2.17[i][A][7] Screenshot E {Letters – Number}

The National School Of Music

Add Student

[Add New Student](#) [Return to Students Table](#)

First Name:	<input type="text"/>	Data Required. Please Complete.
Other Name(s):	<input type="text"/>	Data Required. please Complete.
Surname	<input type="text"/>	Data Required. Please Complete.
Date Of Birth:	14/04/2017	
Address:	<input type="text"/>	Data Required. Please Complete.
Town	<input type="text"/>	Data Required. Please Complete.
PostCode	BT523L3	[PostCode Format = LL000LL] Please Replace final digits with valid characters.
Contact Number:	<input type="text"/>	The Town value entered is incorrect, please enter text.
Email Address	<input type="text"/>	Email Must Contain '@'.
GradeID	1	
InstrumentID	1	
Tuition Fee Received	<input checked="" type="checkbox"/>	

2.17[i][A][7] Screenshot F {Letters – Number}

The screenshot shows an 'Add Student' form with various input fields. Most fields are empty or contain invalid data, resulting in red validation messages:

- First Name:** Data Required. Please Complete.
- Other Name(s):** Data Required. please Complete.
- Surname:** Data Required. Please Complete.
- Date Of Birth:** 14/04/2017 (with a small error icon)
- Address:** Data Required. Please Complete.
- Town:** Data Required. Please Complete.
- PostCode:** BT522EP
- Contact Number:** The Town value entered is incorrect, please enter text.
- Email Address:** Email Must Contain '@'.
- GradeID:** 1
- InstrumentID:** 1
- Tuition Fee Received:**

2.17[i][A][7] Screenshot G {Letters – Symbol}

The screenshot shows an 'Add Student' form with various input fields. Most fields are empty or contain invalid data, resulting in red validation messages:

- First Name:** Data Required. Please Complete.
- Other Name(s):** Data Required. please Complete.
- Surname:** Data Required. Please Complete.
- Date Of Birth:** 14/04/2017 (with a small error icon)
- Address:** Data Required. Please Complete.
- Town:** Data Required. Please Complete.
- PostCode:** BT523L@ (incorrect format)
- Contact Number:** [PostCode Format = LL000LL] Please Replace final digits with valid characters.
- Email Address:** The Town value entered is incorrect, please enter text.
- GradeID:** 1
- InstrumentID:** 1
- Tuition Fee Received:**

2.17[i][A][7] Screenshot H {Letters – Symbol}

The screenshot shows an 'Add Student' form with various input fields and validation messages. The validation messages include:

- First Name: Data Required. Please Complete.
- Other Name(s): Data Required. please Complete.
- Surname: Data Required. Please Complete.
- Date Of Birth: 14/04/2017 (with a small icon next to it)
- Address: Data Required. Please Complete.
- Town: Data Required. Please Complete.
- PostCode: BT522EP
- Contact Number: The Town value entered is incorrect, please enter text.
- Email Address: Email Must Contain '@'.
- GradeID: 1
- InstrumentID: 1
- Tuition Fee Received:

2.17[i][A][7] Screenshot I {Letters – Punctuation}

The screenshot shows an 'Add Student' form with various input fields and validation messages. The validation messages include:

- First Name: Data Required. Please Complete.
- Other Name(s): Data Required. please Complete.
- Surname: Data Required. Please Complete.
- Date Of Birth: 14/04/2017 (with a small icon next to it)
- Address: Data Required. Please Complete.
- Town: Data Required. Please Complete.
- PostCode: BT523L" (Note the extra closing quote)
- Contact Number: [PostCode Format = LL000LL] Please Replace final digits with valid characters.
- Email Address: The Town value entered is incorrect, please enter text.
- GradeID: 1
- InstrumentID: 1
- Tuition Fee Received:

2.17[i][A][7] Screenshot J {Letters – Punctuation }

Add Student

Add New Student **Return to Students Table**

First Name:	<input type="text"/>	Data Required. Please Complete.
Other Name(s):	<input type="text"/>	Data Required. please Complete.
Surname:	<input type="text"/>	Data Required. Please Complete.
Date Of Birth:	<input type="text"/> 14/04/2017	
Address:	<input type="text"/>	Data Required. Please Complete.
Town:	<input type="text"/>	Data Required. Please Complete.
PostCode:	<input type="text"/> BT522EP	
Contact Number:	<input type="text"/>	The Town value entered is incorrect, please enter text.
Email Address:	<input type="text"/>	Email Must Contain '@'.
GradeID:	<input type="text"/> 1	
InstrumentID:	<input type="text"/> 1	
Tuition Fee Received:	<input type="checkbox"/>	

2.17[i][A][7] Screenshot K {Numbers – Letter}

Add Student

Add New Student **Return to Students Table**

First Name:	<input type="text"/>	Data Required. Please Complete.
Other Name(s):	<input type="text"/>	Data Required. please Complete.
Surname:	<input type="text"/>	Data Required. Please Complete.
Date Of Birth:	<input type="text"/> 14/04/2017	
Address:	<input type="text"/>	Data Required. Please Complete.
Town:	<input type="text"/>	Data Required. Please Complete.
PostCode:	<input type="text"/> BT52eLT	[PostCode Format = LL000LL] Please Replace '000' section with valid characters.
Contact Number:	<input type="text"/>	The Town value entered is incorrect, please enter text.
Email Address:	<input type="text"/>	Email Must Contain '@'.
GradeID:	<input type="text"/> 1	
InstrumentID:	<input type="text"/> 1	
Tuition Fee Received:	<input type="checkbox"/>	

2.17[i][A][7] Screenshot L { Numbers – Letter }

Add Student

Add New Student **Return to Students Table**

First Name:	<input type="text"/>	Data Required. Please Complete.
Other Name(s):	<input type="text"/>	Data Required. please Complete.
Surname:	<input type="text"/>	Data Required. Please Complete.
Date Of Birth:	14/04/2017	
Address:	<input type="text"/>	Data Required. Please Complete.
Town:	<input type="text"/>	Data Required. Please Complete.
PostCode:	BT522EP	
Contact Number:	<input type="text"/>	The Town value entered is incorrect, please enter text..
Email Address:	<input type="text"/>	Email Must Contain '@'.
GradeID:	1	
InstrumentID:	1	
Tuition Fee Received:	<input checked="" type="checkbox"/>	

2.17[i][A][7] Screenshot M { Limit }

Add Student

Add New Student **Return to Students Table**

First Name:	<input type="text"/>	Data Required. Please Complete.
Other Name(s):	<input type="text"/>	Data Required. please Complete.
Surname:	<input type="text"/>	Data Required. Please Complete.
Date Of Birth:	14/04/2017	
Address:	<input type="text"/>	Data Required. Please Complete.
Town:	<input type="text"/>	Data Required. Please Complete.
PostCode:	BT43 E	[PostCode Format = LL000LL] Please use full format..
Contact Number:	<input type="text"/>	The Town value entered is incorrect, please enter text..
Email Address:	<input type="text"/>	Email Must Contain '@'.
GradeID:	1	
InstrumentID:	1	
Tuition Fee Received:	<input checked="" type="checkbox"/>	

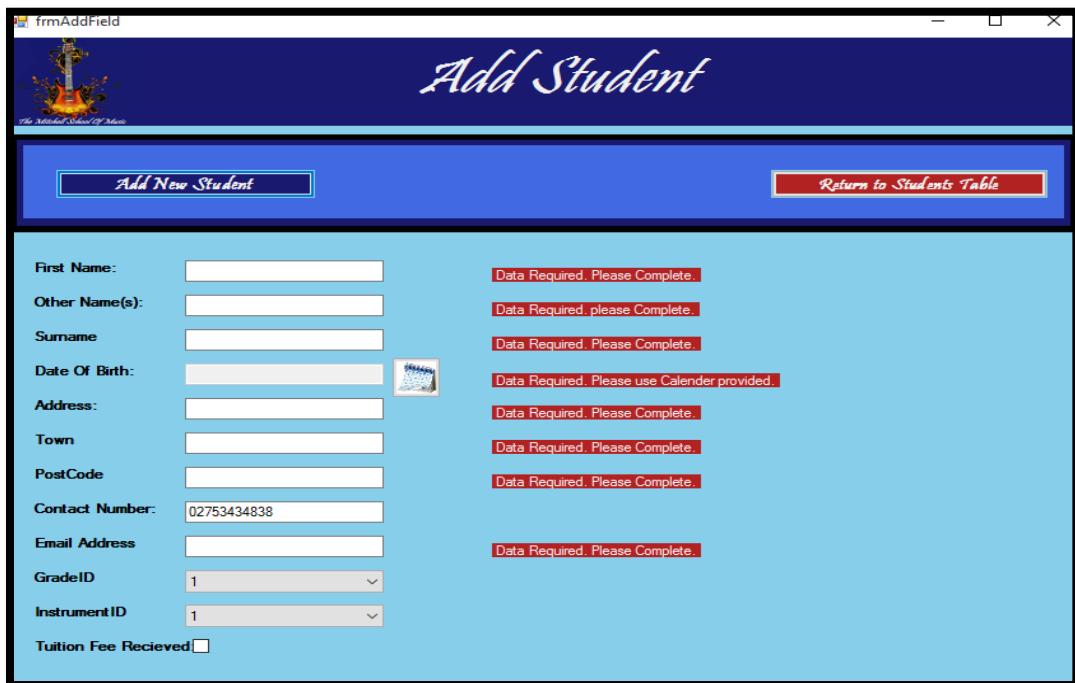
2.17[i][A][7] Screenshot N { Limit }

The screenshot shows the 'Add Student' form from 'The Attic School Of Music'. The form includes fields for First Name, Other Name(s), Surname, Date Of Birth, Address, Town, PostCode, Contact Number, Email Address, GradeID, InstrumentID, and Tuition Fee Received. Most fields have validation errors displayed in red:

- First Name: Data Required. Please Complete.
- Other Name(s): Data Required. please Complete.
- Surname: Data Required. Please Complete..
- Date Of Birth: Data Required. Please use Calendar provided.
- Address: Data Required. Please Complete..
- Town: The Town value entered is incorrect, please enter text.
- Email Address: Email Must Contain '@'.
- GradeID: Data Required. Please Complete.
- InstrumentID: Data Required. Please Complete.
- Tuition Fee Received: Data Required. Please Complete.

2.17[i][A][8] – Contact Number Validation**2.17[i][A][8] Screenshot A {Blank}**

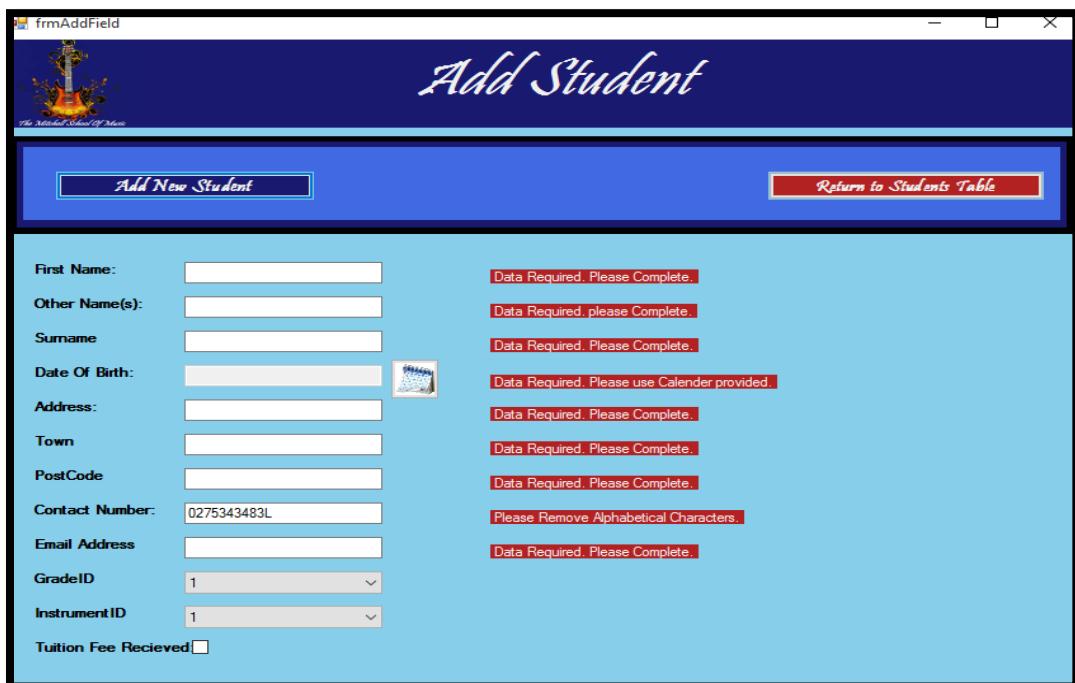
This screenshot of the 'Add Student' form shows the same set of validation errors as the previous one, but with an additional note next to the Date Of Birth field: 'Data Required. Please use Calendar provided.'

2.17[i][A][8] Screenshot B {Blank}


Add Student

Add New Student **Return to Students Table**

First Name:	<input type="text"/>	Data Required. Please Complete.
Other Name(s):	<input type="text"/>	Data Required. please Complete.
Surname	<input type="text"/>	Data Required. Please Complete.
Date Of Birth:	<input type="text"/> 	Data Required. Please use Calender provided.
Address:	<input type="text"/>	Data Required. Please Complete.
Town	<input type="text"/>	Data Required. Please Complete.
PostCode	<input type="text"/>	Data Required. Please Complete.
Contact Number:	<input type="text"/> 0275343483	
Email Address	<input type="text"/>	Data Required. Please Complete.
GradeID	<input type="text"/> 1	
InstrumentID	<input type="text"/> 1	
Tuition Fee Received	<input type="checkbox"/>	

2.17[i][A][8] Screenshot C {Letter Present}


Add Student

Add New Student **Return to Students Table**

First Name:	<input type="text"/>	Data Required. Please Complete.
Other Name(s):	<input type="text"/>	Data Required. please Complete.
Surname	<input type="text"/>	Data Required. Please Complete.
Date Of Birth:	<input type="text"/> 	Data Required. Please use Calender provided.
Address:	<input type="text"/>	Data Required. Please Complete.
Town	<input type="text"/>	Data Required. Please Complete.
PostCode	<input type="text"/>	Data Required. Please Complete.
Contact Number:	<input type="text"/> 0275343483L	Please Remove Alphabetical Characters.
Email Address	<input type="text"/>	Data Required. Please Complete.
GradeID	<input type="text"/> 1	
InstrumentID	<input type="text"/> 1	
Tuition Fee Received	<input type="checkbox"/>	

2.17[i][A][8] Screenshot D {Symbol Present}

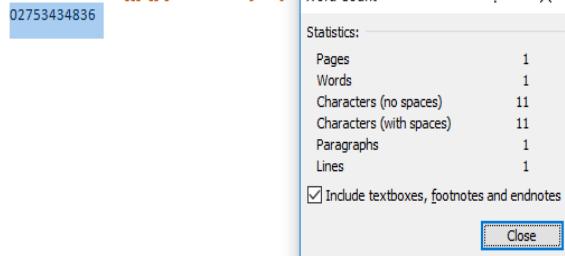
The screenshot shows the 'Add Student' form from the 'frmAddField' application. The form includes fields for First Name, Other Name(s), Surname, Date Of Birth, Address, Town, PostCode, Contact Number, Email Address, GradeID, InstrumentID, and Tuition Fee Received. Most fields have validation errors displayed in red text to the right of the input field. The 'Contact Number' field contains a dollar sign (\$) and has a red error message 'Please Remove symbols.' The 'Email Address' field contains an '@' symbol and has a red error message 'Data Required. Please Complete.' The 'Tuition Fee Received' checkbox has a red error message 'Data Required. Please Complete.' A small icon of a spiral notebook is located next to the Date Of Birth field.

2.17[i][A][8] Screenshot E {Punctuation Present}

The screenshot shows the same 'Add Student' form as in Screenshot D. The validation errors are identical, except for the 'Contact Number' field which now contains an '@' symbol instead of a dollar sign. The red error message 'Please Remove symbols.' remains the same. The other fields (Address, Email Address, and Tuition Fee Received) still show their respective validation errors related to punctuation.

2.17[i][A][8] Screenshot F {Limit}

2.17[i][A][8] Screenshot F {Limit}

2.17[i][A][8] Screenshot G {Limit}**2.17[i][A][9] – Email Address Validation****2.17[i][A][9] Screenshot A {Blank}**

Add Student

Add New Student **Return to Students Table**

First Name: Data Required. Please Complete.

Other Name(s): Data Required. please Complete.

Surname: Data Required. Please Complete.

Date Of Birth: 14/04/2017

Address: Data Required. Please Complete.

Town: Data Required. Please Complete.

PostCode: Data Required. Please Complete.

Contact Number: The Town value entered is incorrect, please enter text.

Email Address: Email Must Contain "@".

GradeID: 1

InstrumentID: 1

Tuition Fee Received:

2.17[i][A][9] Screenshot B {Blank}

frmAddField

The Musical School Of Music

Add Student

[Add New Student](#) [Return to Students Table](#)

First Name:	<input type="text"/>	Data Required. Please Complete..
Other Name(s):	<input type="text"/>	Data Required. please Complete..
Surname:	<input type="text"/>	Data Required. Please Complete..
Date Of Birth:	<input type="text" value="14/04/2017"/>	
Address:	<input type="text"/>	Data Required. Please Complete..
Town:	<input type="text"/>	Data Required. Please Complete..
PostCode:	<input type="text"/>	Data Required. Please Complete..
Contact Number:	<input type="text"/>	The Town value entered is incorrect, please enter text..
Email Address:	<input type="text" value="Test@Test.com"/>	
GradeID	<input type="text" value="1"/>	
InstrumentID	<input type="text" value="1"/>	
Tuition Fee Received	<input type="checkbox"/>	

2.17[i][A][9] Screenshot C {No "@"}

frmAddField

The Musical School Of Music

Add Student

[Add New Student](#) [Return to Students Table](#)

First Name:	<input type="text"/>	Data Required. Please Complete..
Other Name(s):	<input type="text"/>	Data Required. please Complete..
Surname:	<input type="text"/>	Data Required. Please Complete..
Date Of Birth:	<input type="text" value="14/04/2017"/>	
Address:	<input type="text"/>	Data Required. Please Complete..
Town:	<input type="text"/>	Data Required. Please Complete..
PostCode:	<input type="text"/>	Data Required. Please Complete..
Contact Number:	<input type="text"/>	The Town value entered is incorrect, please enter text..
Email Address:	<input type="text" value=""/>	Email Must Contain '@'..
GradeID	<input type="text" value="1"/>	
InstrumentID	<input type="text" value="1"/>	
Tuition Fee Received	<input type="checkbox"/>	

2.17[i][A][9] Screenshot D {No "@"}

The screenshot shows the 'Add Student' form from 'The Mitchell School Of Music'. The form includes fields for First Name, Other Name(s), Surname, Date Of Birth, Address, Town, PostCode, Contact Number, Email Address, GradeID, InstrumentID, and Tuition Fee Received. Most fields have validation errors displayed next to them in red text.

Field	Value Entered	Error Message
First Name		Data Required. Please Complete.
Other Name(s)		Data Required. please Complete.
Surname		Data Required. Please Complete.
Date Of Birth	14/04/2017	(No error message visible)
Address		Data Required. Please Complete.
Town		Data Required. Please Complete.
PostCode		Data Required. Please Complete.
Contact Number		The Town value entered is incorrect, please enter text.
Email Address	Test@Test.com	
GradeID	1	
InstrumentID	1	
Tuition Fee Received	<input type="checkbox"/>	

2.17[i][A][9] Screenshot E {Multiple "@"'s}

The screenshot shows the same 'Add Student' form as in Screenshot D. The Email Address field contains 'Test@Test@.com', which triggers a specific validation error: 'Only one '@' may be present within Email.'

Field	Value Entered	Error Message
First Name		Data Required. Please Complete.
Other Name(s)		Data Required. please Complete.
Surname		Data Required. Please Complete.
Date Of Birth	14/04/2017	(No error message visible)
Address		Data Required. Please Complete.
Town		Data Required. Please Complete.
PostCode		Data Required. Please Complete.
Contact Number		The Town value entered is incorrect, please enter text.
Email Address	Test@Test@.com	Only one '@' may be present within Email.
GradeID	1	
InstrumentID	1	
Tuition Fee Received	<input type="checkbox"/>	

2.17[i][A][9] Screenshot F {Multiple "@"s}

Add Student

[Add New Student](#) [Return to Students Table](#)

First Name:	<input type="text"/>	Data Required. Please Complete.
Other Name(s):	<input type="text"/>	Data Required. please Complete.
Surname:	<input type="text"/>	Data Required. Please Complete.
Date Of Birth:	<input type="text"/> 14/04/2017	
Address:	<input type="text"/>	Data Required. Please Complete.
Town:	<input type="text"/>	Data Required. Please Complete.
PostCode:	<input type="text"/>	Data Required. Please Complete.
Contact Number:	<input type="text"/>	The Town value entered is incorrect, please enter text.
Email Address:	<input type="text"/> Test@Test.com	
GradeID:	<input type="text"/> 1	
InstrumentID:	<input type="text"/> 1	
Tuition Fee Received:	<input type="checkbox"/>	

2.17[i][A][9] Screenshot G {Limit}

Add Student

[Add New Student](#) [Return to Students Table](#)

First Name:	<input type="text"/>	Data Required. Please Complete.
Other Name(s):	<input type="text"/>	Data Required. please Complete.
Surname:	<input type="text"/>	Data Required. Please Complete.
Date Of Birth:	<input type="text"/> 14/04/2017	
Address:	<input type="text"/>	Data Required. Please Complete..
Town:	<input type="text"/>	Data Required. Please Complete..
PostCode:	<input type="text"/>	Data Required. Please Complete..
Contact Number:	<input type="text"/>	The Town value entered is incorrect, please enter text.
Email Address:	<input type="text"/> @@@@@@@@.com	Only one '@' may be present within Email.
GradeID:	<input type="text"/> 1	
InstrumentID:	<input type="text"/> 1	
Tuition Fee Received:	<input type="checkbox"/>	

2.17[i][A][9] Screenshot H {Limit}

The screenshot shows a Windows application window titled "frmAddField". Inside, there are two validation dialogs. The top one is titled "Grade ID Validation" and says "Validation rule: GradeID must contain a valid integer value". The bottom one is titled "Instrument ID Validation" and says "Validation rule: InstrumentID must contain a valid integer value". Both dialogs have "OK" and "Cancel" buttons.

A "Word Count" dialog is overlaid on the application. It displays the following statistics:

Statistics:	
Pages	1
Words	1
Characters (no spaces)	50
Characters (with spaces)	50
Paragraphs	1
Lines	1

Include textboxes, footnotes and endnotes

463

2.17[i][A][10] – Grade ID Validation**2.17[i][A][10] Screenshot A**

The screenshot shows an "Add Student" form titled "frmAddField". The form has a blue header bar with "Add New Student" and "Return to Students Table" buttons. Below the header is a decorative graphic of a guitar.

The form fields and their validation messages are:

- First Name: Data Required. Please Complete.
- Other Name(s): Data Required. please Complete.
- Surname: Data Required. Please Complete.
- Date Of Birth: 14/04/2017 (No validation message)
- Address: Data Required. Please Complete.
- Town: Data Required. Please Complete.
- PostCode: Data Required. Please Complete.
- Contact Number: The Town value entered is incorrect, please enter text..
- Email Address: Only one '@' may be present within Email.
- GradeID: 1 (No validation message)
- InstrumentID: 1
2
3 (Validation message: The Town value entered is incorrect, please enter text.)
- Tuition Fee Received: 4 (No validation message)

2.17[i][A][10] Screenshot B

The screenshot shows the 'Add Student' form with several validation errors displayed in red text next to the input fields:

- First Name: Data Required. Please Complete.
- Other Name(s): Data Required. please Complete.
- Surname: Data Required. Please Complete.
- Date Of Birth: 14/04/2017 (with a calendar icon)
- Address: Data Required. Please Complete.
- Town: Data Required. Please Complete.
- PostCode: Data Required. Please Complete.
- Contact Number: The Town value entered is incorrect, please enter text.
- Email Address: Only one '@' may be present within Email.
- GradeID: 3
- InstrumentID: 3 (with a dropdown arrow)
- Tuition Fee Received:

2.17[i][A][11] – Instrument ID Validation**2.17[i][A][11] Screenshot A**

The screenshot shows the 'Add Student' form with validation errors and a focused dropdown menu for 'InstrumentID':

- First Name: Data Required. Please Complete.
- Other Name(s): Data Required. please Complete.
- Surname: Data Required. Please Complete.
- Date Of Birth: 14/04/2017 (with a calendar icon)
- Address: Data Required. Please Complete.
- Town: Data Required. Please Complete.
- PostCode: Data Required. Please Complete.
- Contact Number: The Town value entered is incorrect , please enter text.
- Email Address: Only one '@' may be present within Email.
- GradeID: 3
- InstrumentID:
 - 3
 - 1
 - 2
 - 3
 - 4 (The option '2' is highlighted with a blue background)
- Tuition Fee Received:

2.17[i][A][11] Screenshot B

Add Student

Add New Student **Return to Students Table**

First Name:	<input type="text"/>	Data Required. Please Complete.
Other Name(s):	<input type="text"/>	Data Required. please Complete.
Surname:	<input type="text"/>	Data Required. Please Complete.
Date Of Birth:	14/04/2017	(calendar icon)
Address:	<input type="text"/>	Data Required. Please Complete.
Town:	<input type="text"/>	Data Required. Please Complete.
PostCode:	<input type="text"/>	Data Required. Please Complete.
Contact Number:	<input type="text"/>	The Town value entered is incorrect, please enter text.
Email Address:	<input type="text"/>	Only one '@' may be present within Email.
GradeID	3	
InstrumentID	2	
Tuition Fee Received	<input type="checkbox"/>	

2.17[i][A][12] – Tuition Fee Validation**2.17[i][A][12] Screenshot A**

Add Student

Add New Student **Return to Students Table**

First Name:	<input type="text"/>	Data Required. Please Complete.
Other Name(s):	<input type="text"/>	Data Required. please Complete.
Surname:	<input type="text"/>	Data Required. Please Complete.
Date Of Birth:	14/04/2017	(calendar icon)
Address:	<input type="text"/>	Data Required. Please Complete.
Town:	<input type="text"/>	Data Required. Please Complete.
PostCode:	<input type="text"/>	Data Required. Please Complete.
Contact Number:	<input type="text"/>	The Town value entered is incorrect, please enter text.
Email Address:	<input type="text"/>	Only one '@' may be present within Email.
GradeID	3	
InstrumentID	2	
Tuition Fee Received	<input type="checkbox"/>	

2.17[i][A][12] Screenshot B

Add Student

[Add New Student](#) [Return to Students Table](#)

First Name: Data Required. Please Complete.

Other Name(s): Data Required. please Complete.

Surname: Data Required. Please Complete.

Date Of Birth: 14/04/2017

Address: Data Required. Please Complete.

Town: Data Required. Please Complete.

PostCode: Data Required. Please Complete.

Contact Number: The Town value entered is incorrect, please enter text.

Email Address: Only one '@' may be present within Email.

GradeID:

InstrumentID:

Tuition Fee Received:

Process [B]) Update Display**2.17[i][B][1] – Update Display Information****2.17[i][B][1] Screenshot A**

Student List

[Search for Student:](#) Adam Ackles

[previous Student Record](#) [Next Student Record](#)

[Add new Student](#) [Update Student Record](#) [Remove Student Record](#)

[Return to Main Menu](#)

StudentID: 4

First Name:	Linda
Other Name(s):	Louise
Surname:	Simmons
Date Of Birth:	20/02/1998
Address:	14 Greenroad
Town:	Coleraine
Post Code:	BT51 2HE
Contact Number:	70359371
Email Address:	Linda854@gmail.com
Grade ID:	2
Instrument ID:	1
Tuition Fee Received:	True

2.17[i][B][1] Screenshot B

The screenshot shows a Windows application window titled "Add Student". The main title bar says "Add Student". Below it, there's a sub-header "Add New Student" and a link "Return to Students Table". The form contains the following fields:

- Student ID:** 4
- First Name:** Linda
- Other Name(s):** Louise
- Surname:** Simmons
- Date Of Birth:** 20/02/1998
- Address:** 14 Greenroad
- Town:** Coleraine
- PostCode:** BT51 2HE
- Contact Number:** 70359371
- Email Address:** Linda854@gmail.com
- GradeID:** 1
- InstrumentID:** 1
- Tuition Fee Received:**

Process [C]) Modifications**2.17[i][C][1] – Add New Student Record****2.17[i][C][1] Screenshot A**

```

    if (ValidFields[x] == true)
    {
        ValidationCount++;
    }

    if (ValidationCount == ValidFields.Count())
    {
        AddNewInformation();
    }
}

// AddNewInformation()
{
    if (GlobalVariables.PreviousForm == "StudentTable")
    {
        if (GlobalVariables.Purpose == "Add")
        {
            QueryString = "INSERT INTO Students VALUES(@First_Name, @Other_Names, @Surname, @DOB, @Address, @Town, @Post, @Contact, @Email, @Grade, @Instrument, @Tuition)";
        }
        else
        {
            QueryString = "UPDATE Students SET First_Name=@First_Name, Other_Names=@Other_Names, Surname=@Surname, Date_of_Birth=@DOB, Address=@Address, Town=@Town, PostCode=@Post, Contact_Number=@Contact, Email_Address=@Email, GradeID=@Grade, InstrumentID=@Instrument, Tuition_Fee=@Tuition WHERE StudentID=@StudentID";
        }
    }

    using (connection = new SqlConnection(connectionString))
    {
        connection.Open();
        using (SqlCommand Command = new SqlCommand(QueryString, connection))
        using (SqlParameter[] Parameters = new SqlParameter[11])
        {
            Parameters.AddWithValue("@StudentID", tbFieldInfo14.Text);
            Parameters.AddWithValue("@First_Name", tbFieldInfo1.Text);
            Parameters.AddWithValue("@Other_Names", tbFieldInfo10.Text);
            Parameters.AddWithValue("@Surname", tbFieldInfo2.Text);
            Parameters.AddWithValue("@DOB", tbFieldInfo3.Text);
            Parameters.AddWithValue("@Address", tbFieldInfo4.Text);
            Parameters.AddWithValue("@Town", tbFieldInfo5.Text);
            Parameters.AddWithValue("@Post", tbFieldInfo6.Text);
            Parameters.AddWithValue("@Contact", tbFieldInfo7.Text);
            Parameters.AddWithValue("@Email", tbFieldInfo8.Text);
            Parameters.AddWithValue("@Grade", tbFieldInfo9.Text);
            Parameters.AddWithValue("@Instrument", tbFieldInfo10.Text);
            Parameters.AddWithValue("@Tuition", tbFieldInfo11.Text);

            Command.Parameters.AddWithValue("@First_Name", tbFieldInfo1.Text);
            Command.Parameters.AddWithValue("@Other_Names", tbFieldInfo10.Text);
            Command.Parameters.AddWithValue("@Surname", tbFieldInfo2.Text);
            Command.Parameters.AddWithValue("@DOB", tbFieldInfo3.Text);
            Command.Parameters.AddWithValue("@Address", tbFieldInfo4.Text);
            Command.Parameters.AddWithValue("@Town", tbFieldInfo5.Text);
            Command.Parameters.AddWithValue("@Post", tbFieldInfo6.Text);
            Command.Parameters.AddWithValue("@Contact", tbFieldInfo7.Text);
            Command.Parameters.AddWithValue("@Email", tbFieldInfo8.Text);
            Command.Parameters.AddWithValue("@Grade", tbFieldInfo9.Text);
            Command.Parameters.AddWithValue("@Instrument", tbFieldInfo10.Text);
            Command.Parameters.AddWithValue("@Tuition", tbFieldInfo11.Text);

            Command.ExecuteNonQuery();
        }
        connection.Close();
    }
}

```

2.17[i][C][1] Screenshot B

Add New Student **Return to Students Table**

First Name:

Other Name(s):

Surname:

Date Of Birth: 

Address:

Town:

PostCode:

Contact Number:

Email Address:

GradeID:

InstrumentID:

Tuition Fee Received

2.17[i][C][1] Screenshot C

	StudentID	First_Name	Other_Names	Surname	DateOfBirth	Address	Town	PostCode	ContactNum...	EmailAddress	GradeID	InstrumentID	Tuition_Fee_R
4	LindaT	Louise		Simmons	20/02/1998	14 Greenroad	Coleraine	BT51 2HE	70359371	Linda54@gmail...	3	3	False
5	Trevor	Luke		Simmons	20/08/1997	22 The Glebe	Ballymoney	BT32 3ND	74920033	Trevorsimmons...	4	4	True
6	Daniel	Robert		Waters	01/02/1970	199 Pacroad	Ballymoney	BT47 3MF	37719048	danielwaters99...	4	3	True
7	Adam	Brent	Ackles		30/06/1993	731 boastlane	Garragh	BT43 2HT	46299567	adambright@ya...	2	2	False
8	Jenson	James	Tweed		22/12/1990	71 glebe avenue	Ballycastle	BT37 5GW	3392321	jtwedd@hotmail...	4	5	True
9	Diana	Ann	James		14/04/1985	6 Polyroad	Bushmills	BT83 5FE	40039237	dianajames@ya...	4	6	True
10	Lorraine	Hannah	Silvers		17/08/1999	48 union street	Postrush	BT84 2TR	4002784	loriane@silvers...	1	7	False
11	Courtney	Lisa	Doherty		13/01/1994	11 jacobsLane	Garragh	BT95 3RE	47326678	courtney20@q...	1	8	True
12	Patrick	Samuel	Roberts		20/02/1973	92 Ballyway	Ballybofey	BT05 5RF	30283736	patrick26@yahoo...	2	9	True
13	Matthew	Nathan	Jones		26/05/1975	45 tinkers street	Portstewart	BT82 7EQ	46335362	matthewjones...	3	2	False
14	Samantha	Alison	Mc farlane		17/05/1982	231 Anderson A...	Ballymoney	BT90 3UN	45672623	sammym4@yahoo...	2	4	True
15	Linda	Hayley	Spears		04/04/1996	999 Goldhill	Ballymena	BT71 9YR	37434794	lindasppear29...	3	5	False
16	Lucy	Susan	Dysart		28/07/1991	52 Colour avenue	Coleraine	BT52 1QA	50382627	tonyhoward@q...	4	6	True
17	Anthony	Ryan	Howard		12/03/1990	32 Hillview	Postrush	BT95 3YU	50947464	tonyhoward@q...	3	10	True
18	Cleo	Laura	Kane		23/02/2000	32 Hillview	Ballybofey	BT59 2EC	3892374	cleopatra6@q...	4	2	True
19	Louise	Amanda	Doran		04/09/1994	39 Union street	Coleraine	BT51 3EH	75434924	loulobodoran009...	3	5	True
20	Gerard	Josh	Mc daid		25/04/1968	23 Chief Street	Bushmills	BT78 4YT	46637328	gerard300@gm...	1	7	False
21	Kayla	Brooke	Black		13/10/1978	91 butchroad	Ballymoney	BT94 2VU	53752852	kaylablack@ya...	2	6	False
22	Kristeen	Casey	Dickerson		14/11/1979	19 Screenhill ro...	BallyCastle	BT39 5GF	53278523	Kdickerson@Y...	2	10	False
23	Leanna	Skylar	Philips		01/01/1992	124 Strand Park	Garragh	BT43 9KV	96854746	LeaSky42@Hot...	1	3	False
24	Duke	Kassy	Ewart		25/12/1998	220 Anderson A...	Postrush	BT59 4RM	5732562	DukeEwart@G...	1	3	False
25	Alan	Shantelle	Blue		21/08/2995	60 Tullyarton D...	BallyBofey	BT99 0CI	53275823	AlanBlue@Yah...	4	2	True
26	Reginald	Jessa	Stevenson		05/07/1982	14 Gelbe Park	Coleraine	BT46 2HD	13123554	reg24@gmail.co...	1	9	False
27	Fredrica	Trevelyn	Cobb		09/03/2000	84 Crescent Road	Coleraine	BT59 5GL	70328356	FredCobb@Hot...	3	4	True
28	Test	Test	Test		12/04/2017	123 Test	Test	bt543ew	4444444444	test@	1	1	False
29													
30													
31													

2.17[i][C][2] – Update old Student Record**2.17[i][C][2] Screenshot A**

```

    if (ValidationCount == ValidFields.Count())
    {
        AddNewInformation();
    }
}

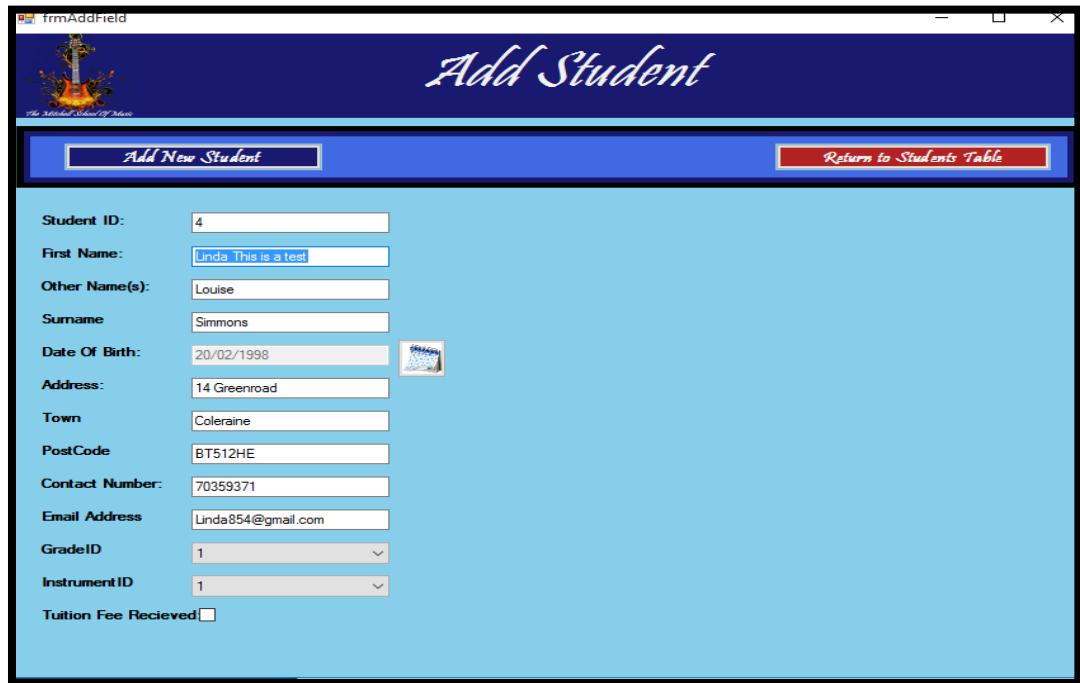
public void AddNewInformation()
{
    if (GlobalVariables.PreviousForm == "StudentTable")
    {
        if (GlobalVariables.Purpose == "Add")
        {
            QueryString = "INSERT INTO Students VALUES(@First_Name, @Other_Names, @Surname, @DOB, @Address, @Town, @Post, @Contact, @Email, @Grade, @Instrument, @Tuition)";
        }
        else
        {
            QueryString = "UPDATE Students SET First_Name=@First_Name, Other_Names=@Other_Names, Surname=@Surname, DateOfBirth=@DOB, Address=@Address, Town=@Town, PostCode=@Post, Contact_Number=@Contact, Email_Address=@Email, GradeID=@Grade, InstrumentID=@Instrument, Tuition=@Tuition WHERE StudentID=@StudentID";
        }
    }

    using (connection = new SqlConnection(connectionString))
    {
        connection.Open();
        using (SqlCommand Command = new SqlCommand(QueryString, connection))
        {
            Command.Parameters.AddWithValue("@StudentID", tbFieldInfo1.Text);
            Command.Parameters.AddWithValue("@First_Name", tbFieldInfo2.Text);
            Command.Parameters.AddWithValue("@Other_Names", tbFieldInfo3.Text);
            Command.Parameters.AddWithValue("@Surname", tbFieldInfo4.Text);
            Command.Parameters.AddWithValue("@DOB", Convert.ToDateTime(tbFieldInfo5.Text));
            Command.Parameters.AddWithValue("@Address", tbFieldInfo6.Text);
            Command.Parameters.AddWithValue("@Town", tbFieldInfo7.Text);
            Command.Parameters.AddWithValue("@Post", tbFieldInfo8.Text);
            Command.Parameters.AddWithValue("@Contact", tbFieldInfo9.Text);
            Command.Parameters.AddWithValue("@Email", tbFieldInfo10.Text);
            Command.Parameters.AddWithValue("@Grade", Convert.ToInt16(cbFieldInfo11.Text));
            Command.Parameters.AddWithValue("@Instrument", Convert.ToInt32(cbFieldInfo12.Text));
            Command.Parameters.AddWithValue("@Tuition", CheckBox1.Text);

            Command.ExecuteNonQuery();
        }
        connection.Close();
    }
}

else if (GlobalVariables.PreviousForm == "TeacherTable")
{
    if (GlobalVariables.Purpose == "Add")
    {
    }
    else
}

```

2.17[i][C][2] Screenshot B

2.17[i][C][2] Screenshot C

StudentID	First_Name	Other_Names	Surname	DateOfBirth	Address	Town	PostCode	Contact_Number	Email_Address
4	Linda	This is a test	Louise	Simmons	20/02/1998	14 Greenroad	Coleraine	BT512HE	70359371
5	Trevor		Luke	Simmons	20/08/1997	22 The Glebe	Ballymoney	BT32 3ND	74920033
6	Daniel		Robert	Waters	01/02/1970	199 Pacroad	Ballymoney	BT47 3MF	37719048
7	Adam		Brent	Ackles	30/06/1993	731 boatlane	Garvagh	BT43 2HT	46299567
8	Jenson		James	Tweed	22/12/1990	71 glebe avenue	Ballycastle	BT37 5GW	33923231
9	Diana		Ann	James	14/04/1985	6 Polyroad	Bushmills	BT83 5FE	40039237
10	Lorraine		Hannah	Silvers	17/08/1999	48 union street	Portrush	BT84 2TR	40027364
11	Courtney		Lisa	Doherty	13/01/1994	11 jacobsLane	Garvagh	BT95 3RE	47326678
12	Patrick		Samuel	Roberts	20/02/1973	92 Ballyway	Ballybogey	BT05 5RF	30283736
13	Matthew		Nathan	Jones	26/06/1975	45 tinkers street	Portstewart	BT82 7EQ	46335362
14	Samantha		Alison	Mc farlane	17/05/1982	231 Anderson A...	Ballymoney	BT00 3JN	45673623
15	Linda		Hayley	Spears	04/04/1996	999 Goldhill	Ballymena	BT71 9YR	37434794
16	Lucy		Susan	Dysart	28/07/1991	52 Colour avenue	Coleraine	BT52 1QA	50382627
17	Anthony		Ryan	Howard	12/03/1990	32 Hillview	Portrush	BT95 3YU	50947464
18	Cleo		Laura	Kane	23/02/2000	32 Hillview	Ballybogey	BT59 2EC	3892374
19	Louise		Amanda	Doran	04/09/1994	39 Union street	Coleraine	BT51 3EH	75434934
20	Gerard		Josh	Mc daid	25/04/1968	23 Chief Street	Bushmills	BT78 4YT	46637328
21	Kayla		Brooke	Black	13/10/1978	91 butchroad	Ballymoney	BT94 2YU	53752852
22	Kristeen		Casey	Dickenson	14/11/1979	19 Screenhill ro...	BallyCastle	BT39 5GF	53278523
23	Leanna		Skyler	Philips	01/01/1992	124 Strand Park	Garvagh	BT43 9KV	96854746
24	Duke		Kassy	Ewart	25/12/1998	220 Anderson A...	Portrush	BT59 4RM	57325632
25	Alan		Shantelle	Blue	21/08/1995	60 Tullyerton D...	BallyBogey	BT99 0CJ	53275823
26	Reginald		Jessa	Stevenson	05/07/1982	14 Gelbe Park	Coleraine	BT46 2HD	13123534
27	Frederica		Trevelyan	Cobb	09/03/2000	84 Crescent Road	Coleraine	BT59 5GL	70329356
31	Test		Test	Test	12/04/2017	123 Test	Test	bt549ew	444444444444
o									test@
	NULL		NULL	NULL	NULL	NULL	NULL	NULL	NULL

2.17[ii] Teacher Record**Process [A]) Validation****2.17[ii][A][1] – First Name Validation****2.17[ii][A][1] Screenshot A {Blank}**

The screenshot shows a web application window titled "Add Teacher". At the top left is the logo of "The National School Of Music". Below the title, there are two buttons: "Add New Teacher" on the left and "Return to Teacher Table" on the right.

The form consists of several input fields:

- First Name:** An input field with a red error message: "Data Required. Please Complete."
- Surname:** An input field with a red error message: "Data Required. Please Complete."
- Address:** An input field with a red error message: "Data Required. Please Complete."
- Town:** An input field with a red error message: "Data Required. Please Complete."
- PostCode:** An input field with a red error message: "Data Required. Please Complete."
- Email Address:** An input field with a red error message: "Email Must Contain '@'."
- Contact Number:** An input field with a red error message: "The Town value entered is incorrect, please enter text."
- Specialisation:** A dropdown menu set to "Brass".
- RoomID:** A dropdown menu set to "1".

2.17[ii][A][i] Screenshot B {Blank}

The screenshot shows a Windows application window titled "frmAddField". The main title bar says "Add Teacher". The window contains several text input fields and dropdown menus. The "First Name" field is populated with "Test". The "Surname" field is empty. The "Address" field is empty and has a red validation message: "Data Required. Please Complete.". The "Town" field is empty and has a red validation message: "Data Required. Please Complete.". The "PostCode" field is empty and has a red validation message: "Data Required. Please Complete.". The "Email Address" field is empty and has a red validation message: "Email Must Contain '@'.". The "Contact Number" field is empty and has a red validation message: "The Town value entered is incorrect, please enter text.". The "Specialisation" dropdown menu is set to "Brass". The "RoomID" dropdown menu is set to "1".

2.17[ii][A][i] Screenshot C {Number}

The screenshot shows the same Windows application window as in Screenshot B. The "First Name" field now contains "Test2". The "Surname" field is empty. The "Address" field is empty and has a red validation message: "Data Required. Please Complete.". The "Town" field is empty and has a red validation message: "Data Required. Please Complete.". The "PostCode" field is empty and has a red validation message: "Data Required. Please Complete.". The "Email Address" field is empty and has a red validation message: "Email Must Contain '@'.". The "Contact Number" field is empty and has a red validation message: "The Town value entered is incorrect, please enter text.". The "Specialisation" dropdown menu is set to "Brass". The "RoomID" dropdown menu is set to "1".

2.17[ii][A][i] Screenshot D {Number}

Add Teacher

Add New Teacher **Return to Teacher Table**

First Name:	Test	
Surname:		Data Required. Please Complete.
Address:		Data Required. Please Complete.
Town:		Data Required. Please Complete.
PostCode:		Email Must Contain '@'.
Email Address:		The Town value entered is incorrect, please enter text.
Contact Number:		
Specialisation:	Brass	
RoomID:	1	

2.17[ii][A][i] Screenshot E {Symbol}

Add Teacher

Add New Teacher **Return to Teacher Table**

First Name:	Test@	Please Remove Invalid Characters. Name should only contain Letters!
Surname:		
Address:		Data Required. Please Complete.
Town:		Data Required. Please Complete.
PostCode:		Data Required. Please Complete.
Email Address:		Email Must Contain '@'.
Contact Number:		The Town value entered is incorrect, please enter text.
Specialisation:	Brass	
RoomID:	1	

2.17[ii][A][i] Screenshot F {Symbol}

The screenshot shows a Windows application window titled "frmAddField" with a dark blue header bar. The title bar features a logo of a guitar and the text "The Attic And School Of Music". The main title "Add Teacher" is centered in a white font. Below the title, there are two buttons: "Add New Teacher" on the left and "Return to Teacher Table" on the right. The form contains several input fields and validation messages:

Field	Value	Error Message
First Name:	Test	
Surname:		Data Required. Please Complete.
Address:		Data Required. Please Complete.
Town:		Data Required. Please Complete.
PostCode:		Data Required. Please Complete.
Email Address:		Email Must Contain '@'.
Contact Number:		The Town value entered is incorrect, please enter text.
Specialisation:	Brass	
RoomID:	1	

2.17[ii][A][i] Screenshot G {Punctuation}

The screenshot shows the same Windows application window as in Screenshot F. The "First Name:" field contains the value "Test)". A red validation message "Please Remove Invalid Characters. Name should only contain Letters!" is displayed next to the field. All other fields and their validation messages remain the same as in Screenshot F.

Field	Value	Error Message
First Name:	Test)"	Please Remove Invalid Characters. Name should only contain Letters!
Surname:		Data Required. Please Complete.
Address:		Data Required. Please Complete.
Town:		Data Required. Please Complete.
PostCode:		Data Required. Please Complete.
Email Address:		Email Must Contain '@'.
Contact Number:		The Town value entered is incorrect, please enter text.
Specialisation:	Brass	
RoomID:	1	

2.17[ii][A][i] Screenshot H {Punctuation}

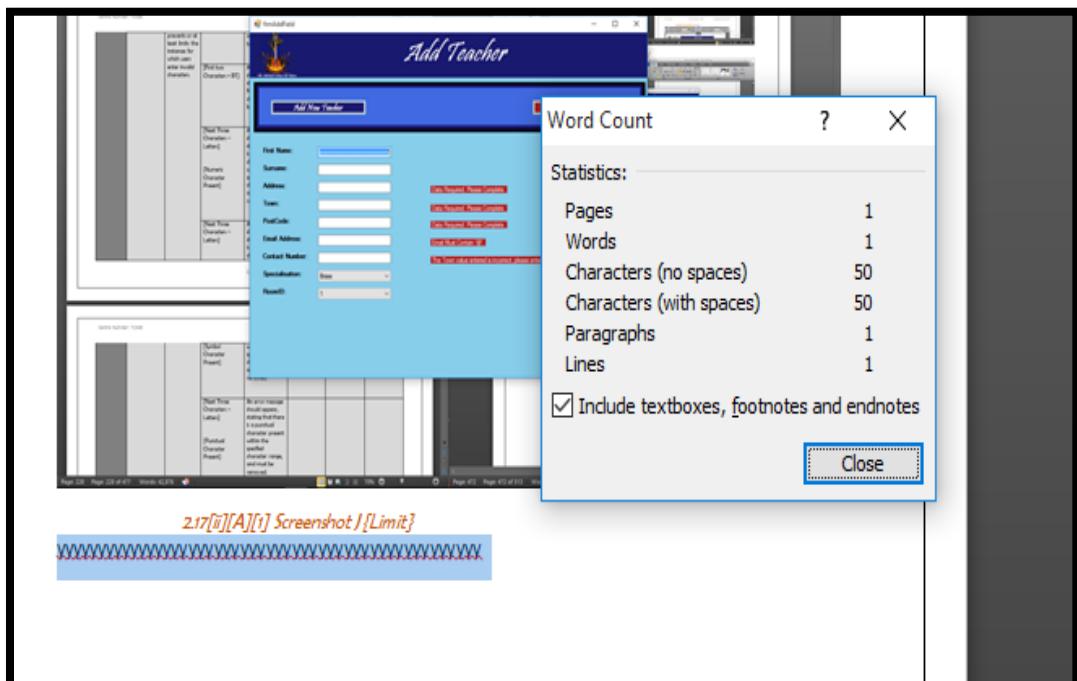
The screenshot shows a Windows application window titled "frmAddField" with a dark blue header bar. The title bar features a logo of a guitar and the text "The National School Of Music". The main title "Add Teacher" is centered in a large, white, cursive font. Below the title is a sub-header "Add New Teacher" in a smaller blue-bordered box and a red "Return to Teacher Table" button.

The form contains the following fields:

- First Name: Test
- Surname: (empty)
- Address: (empty) - Validation message: Data Required. Please Complete.
- Town: (empty) - Validation message: Data Required. Please Complete.
- PostCode: (empty) - Validation message: Data Required. Please Complete.
- Email Address: (empty) - Validation message: Email Must Contain '@'.
- Contact Number: (empty) - Validation message: The Town value entered is incorrect, please enter text.
- Specialisation: Brass
- RoomID: 1

2.17[ii][A][i] Screenshot I {Limit}

This screenshot is identical to Screenshot H, showing the same Windows application window titled "frmAddField". The layout, fields, and validation messages are the same, including the validation message for the "Town" field: "The Town value entered is incorrect, please enter text."

2.17[ii][A][1] Screenshot J {Limit}**2.17[ii][A][i] Screenshot J {Limit}**

www.bbc.co.uk/schools/gcsebitesize/english/poetry/medieval.htm

2.17[ii][A][2] – Surname Validation**2.17[ii][A][2] Screenshot A {Blank}**

A screenshot of a Windows application window titled "frmAddField" with a blue header bar containing the title "Add Teacher". The main area is a form titled "Add New Teacher" with the following fields and validation messages:

Field	Value	Validation Message
First Name:	[Empty]	Data Required. Please Complete.
Surname:	[Empty]	Data Required. Please Complete.
Address:	[Empty]	Data Required. Please Complete.
Town:	[Empty]	Data Required. Please Complete.
PostCode:	[Empty]	Data Required. Please Complete.
Email Address:	[Empty]	Email Must Contain '@'.
Contact Number:	[Empty]	The Town value entered is incorrect, please enter text.
Specialisation:	Brass	
RoomID:	1	

2.17[ii][A][2] Screenshot B {Blank}

The screenshot shows a Windows application window titled "frmAddField". The main title bar says "Add Teacher". Below the title bar is a logo of a guitar and the text "The Musical School Of Music". There are two buttons at the top: "Add New Teacher" on the left and "Return to Teacher Table" on the right. The main area contains eight input fields with validation messages:

Field	Value	Validation Message
First Name:		Data Required. Please Complete.
Surname:	Test	
Address:		Data Required. Please Complete.
Town:		Data Required. Please Complete.
PostCode:		Data Required. Please Complete.
Email Address:		Email Must Contain '@'.
Contact Number:		Data Required. Please Complete.
Specialisation:	Brass	
RoomID:	1	

2.17[ii][A][2] Screenshot C {Number}

The screenshot shows the same Windows application window as in Screenshot B. The validation messages have changed:

Field	Value	Validation Message
First Name:		Data Required. Please Complete.
Surname:	Test2	Please Remove Invalid Characters. Name should only contain Letters!
Address:		Data Required. Please Complete.
Town:		Data Required. Please Complete.
PostCode:		Data Required. Please Complete.
Email Address:		Email Must Contain '@'.
Contact Number:		Data Required. Please Complete.
Specialisation:	Brass	
RoomID:	1	

2.17[ii][A][2] Screenshot D {Number}

Add Teacher

Add New Teacher **Return to Teacher Table**

First Name: Data Required. Please Complete.

Surname: Data Required. Please Complete.

Address: Data Required. Please Complete.

Town: Data Required. Please Complete.

PostCode: Data Required. Please Complete.

Email Address: Email Must Contain '@'.

Contact Number: Data Required. Please Complete.

Specialisation: ▾

RoomID: ▾

2.17[ii][A][2] Screenshot E {Symbol}

Add Teacher

Add New Teacher **Return to Teacher Table**

First Name: Data Required. Please Complete.

Surname: Please Remove Invalid Characters. Name should only contain Letters!

Address: Data Required. Please Complete.

Town: Data Required. Please Complete.

PostCode: Data Required. Please Complete.

Email Address: Email Must Contain '@'.

Contact Number: Data Required. Please Complete.

Specialisation: ▾

RoomID: ▾

2.17[ii][A][2] Screenshot F {Symbol}

The screenshot shows a Windows application window titled "frmAddField". The main title is "Add Teacher". Below it is a logo for "The Attic And School Of Music" featuring a stylized orange electric guitar. The interface includes two buttons: "Add New Teacher" (blue border) and "Return to Teacher Table" (red border). The form contains eight text input fields and their corresponding validation messages:

Field	Value	Error Message
First Name:		Data Required. Please Complete.
Surname:	Test	
Address:		Data Required. Please Complete.
Town:		Data Required. Please Complete.
PostCode:		Data Required. Please Complete.
Email Address:		Email Must Contain '@'.
Contact Number:		Data Required. Please Complete.
Specialisation:	Brass	
RoomID:	1	

2.17[ii][A][2] Screenshot G {Punctuation}

The screenshot shows the same Windows application window as Screenshot F. The validation errors have changed to reflect punctuation issues:

Field	Value	Error Message
First Name:		Data Required. Please Complete.
Surname:	Test!"	Please Remove Invalid Characters. Name should only contain Letters!
Address:		Data Required. Please Complete.
Town:		Data Required. Please Complete.
PostCode:		Data Required. Please Complete.
Email Address:		Email Must Contain '@'.
Contact Number:		Data Required. Please Complete.
Specialisation:	Brass	
RoomID:	1	

2.17[ii][A][2] Screenshot H {Punctuation}

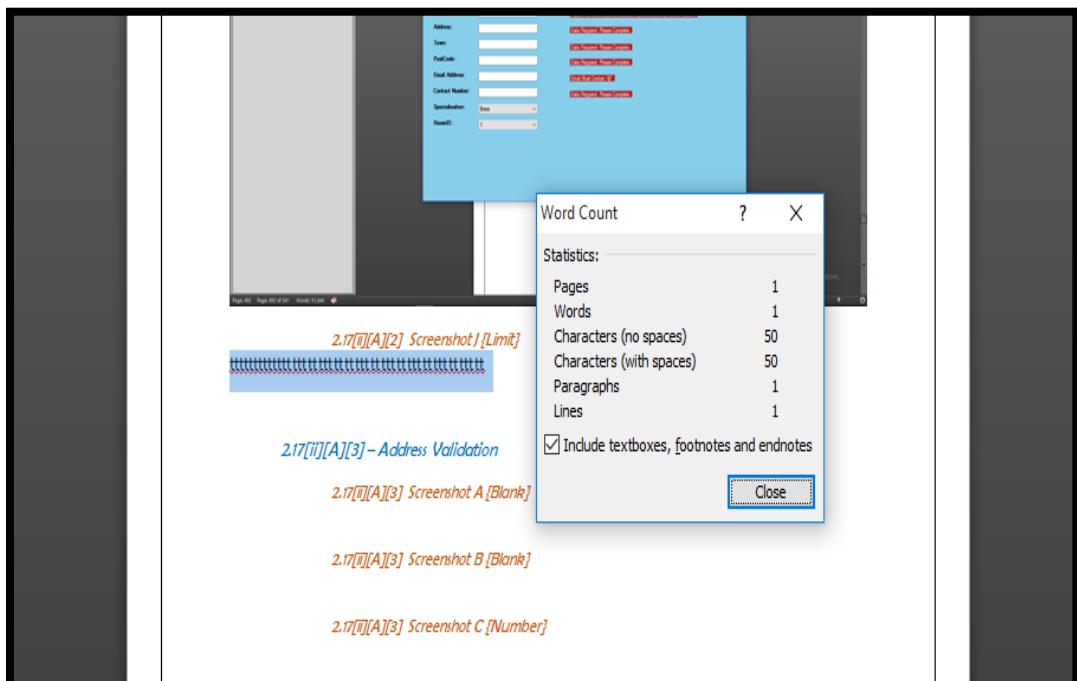
The screenshot shows a Windows application window titled "frmAddField" with a dark blue header bar. The title bar includes the window icon, the title "frmAddField", and standard window control buttons. Below the header is a decorative graphic of a guitar and the text "The Musical School Of Music". The main area is titled "Add Teacher" in a large, stylized font. At the top left is a button labeled "Add New Teacher" and at the top right is a button labeled "Return to Teacher Table". The form contains the following fields:

Label	Value	Error Message
First Name:		Data Required. Please Complete.
Surname:	Test	
Address:		Data Required. Please Complete.
Town:		Data Required. Please Complete.
PostCode:		Data Required. Please Complete.
Email Address:		Email Must Contain '@'.
Contact Number:		Data Required. Please Complete.
Specialisation:	Brass	
RoomID:	1	

2.17[ii][A][2] Screenshot I {Limit}

This screenshot is identical to Screenshot H in layout and field structure, but the validation message for the "Surname" field is different. The "Surname" field contains non-alphabetic characters, and the error message is "Please Remove Invalid Characters. Name should only contain Letters!"

Label	Value	Error Message
First Name:		Data Required. Please Complete.
Surname:	1234567890	Please Remove Invalid Characters. Name should only contain Letters!
Address:		Data Required. Please Complete.
Town:		Data Required. Please Complete.
PostCode:		Data Required. Please Complete.
Email Address:		Email Must Contain '@'.
Contact Number:		Data Required. Please Complete.
Specialisation:	Brass	
RoomID:	1	

2.17[ii][A][2] Screenshot J {Limit}**2.17[ii][A][3] – Address Validation****2.17[ii][A][3] Screenshot A [Blank]**

frmAddField

Add Teacher

Add New Teacher **Return to Teacher Table**

First Name:	<input type="text"/>	Data Required. Please Complete.
Surname:	<input type="text"/>	Data Required. Please Complete.
Address:	<input type="text"/>	Data Required. Please Complete.
Town:	<input type="text"/>	Data Required. Please Complete.
PostCode:	<input type="text"/>	Data Required. Please Complete.
Email Address:	<input type="text"/>	Email Must Contain "@".
Contact Number:	<input type="text"/>	Data Required. Please Complete.
Specialisation:	Brass	
RoomID:	1	

2.17[ii][A][3] Screenshot B {Blank}

The screenshot shows the 'Add Teacher' form with validation errors for all fields. The validation messages are:

- First Name: Data Required. Please Complete.
- Surname: Data Required. Please Complete.
- Address: 123 test (Validation message: Data Required. Please Complete.)
- Town: (Validation message: Data Required. Please Complete.)
- PostCode: (Validation message: Data Required. Please Complete.)
- Email Address: (Validation message: Email Must Contain '@'.)
- Contact Number: (Validation message: Data Required. Please Complete.)
- Specialisation: Brass (Validation message: Data Required. Please Complete.)
- RoomID: 1 (Validation message: Data Required. Please Complete.)

2.17[ii][A][3] Screenshot C {Street Number - Letter}

The screenshot shows the 'Add Teacher' form with validation errors for all fields. The validation messages are:

- First Name: Data Required. Please Complete.
- Surname: Data Required. Please Complete.
- Address: 123e test (Validation message: Please Remove alphabetical Characters from street number.)
- Town: (Validation message: Data Required. Please Complete.)
- PostCode: (Validation message: Data Required. Please Complete.)
- Email Address: (Validation message: Email Must Contain '@'.)
- Contact Number: (Validation message: Data Required. Please Complete.)
- Specialisation: Brass (Validation message: Data Required. Please Complete.)
- RoomID: 1 (Validation message: Data Required. Please Complete.)

2.17[ii][A][3] Screenshot D {Street Number - Symbol}

The screenshot shows the 'Add Teacher' form from 'The Musical School Of Music'. The address field contains '123@ test'. The validation message 'Please Remove Symbol or punctuation.' is displayed next to it.

Field	Value	Error Message
First Name:		Data Required. Please Complete.
Surname:		Data Required. Please Complete.
Address:	123@ test	Please Remove Symbol or punctuation.
Town:		Data Required. Please Complete.
PostCode:		Data Required. Please Complete.
Email Address:		Email Must Contain '@'.
Contact Number:		Data Required. Please Complete.
Specialisation:	Brass	
RoomID:	1	

2.17[ii][A][3] Screenshot E { Street Number - Punctuation }

The screenshot shows the 'Add Teacher' form from 'The Musical School Of Music'. The address field contains '123" test'. The validation message 'Please Remove Symbol or punctuation.' is displayed next to it.

Field	Value	Error Message
First Name:		Data Required. Please Complete.
Surname:		Data Required. Please Complete.
Address:	123" test	Please Remove Symbol or punctuation.
Town:		Data Required. Please Complete.
PostCode:		Data Required. Please Complete.
Email Address:		Email Must Contain '@'.
Contact Number:		Data Required. Please Complete.
Specialisation:	Brass	
RoomID:	1	

2.17[ii][A][3] Screenshot F {Street Name - Number}

The screenshot shows a Windows application window titled "frmAddField". The main title bar says "Add Teacher". Below the title bar is a logo of a guitar with the text "The Musical School Of Music". The window has a blue header bar with "Add New Teacher" and "Return to Teacher Table" buttons. The main area contains fields for First Name, Surname, Address, Town, PostCode, Email Address, Contact Number, Specialisation, and RoomID. The "Address" field contains "123 test4". Error messages are displayed next to the fields: "Data Required. Please Complete." for First Name, Surname, and PostCode; "Please Remove Numerical Characters from street name." for Address; "Data Required. Please Complete." for Town; "Email Must Contain '@'." for Email Address; "Data Required. Please Complete." for Contact Number; and "Data Required. Please Complete." for RoomID.

2.17[ii][A][3] Screenshot G {Street Name - Symbol}

The screenshot shows the same Windows application window as in Screenshot F. The main title bar says "Add Teacher". Below the title bar is the same logo. The window has a blue header bar with "Add New Teacher" and "Return to Teacher Table" buttons. The main area contains fields for First Name, Surname, Address, Town, PostCode, Email Address, Contact Number, Specialisation, and RoomID. The "Address" field contains "123 test@". Error messages are displayed next to the fields: "Data Required. Please Complete." for First Name, Surname, and PostCode; "Please Remove Symbol or punctuation." for Address; "Data Required. Please Complete." for Town; "Data Required. Please Complete." for Email Address; "Data Required. Please Complete." for Contact Number; and "Data Required. Please Complete." for RoomID.

2.17[ii][A][3] Screenshot H { Street Name - Punctuation }

frmAddField

Add Teacher

Add New Teacher *Return to Teacher Table*

First Name:	<input type="text"/>	Data Required. Please Complete.
Surname:	<input type="text"/>	Data Required. Please Complete.
Address:	<input style="border: 2px solid red; color: red; background-color: #f0f0f0; font-style: italic; font-weight: bold; font-size: 1em; padding: 2px; margin: 2px; width: 150px; height: 20px; border-radius: 5px; outline: none;" type="text" value="123 test"/>	Please Remove Symbol or punctuation.
Town:	<input type="text"/>	Data Required. Please Complete.
PostCode:	<input type="text"/>	Data Required. Please Complete.
Email Address:	<input type="text"/>	Email Must Contain '@'.
Contact Number:	<input type="text"/>	Data Required. Please Complete.
Specialisation:	<input style="border: 1px solid black; color: black; background-color: white; font-style: italic; font-weight: bold; font-size: 1em; padding: 2px; margin: 2px; width: 150px; height: 20px; border-radius: 5px; outline: none;" type="text" value="Brass"/>	
RoomID:	<input style="border: 1px solid black; color: black; background-color: white; font-style: italic; font-weight: bold; font-size: 1em; padding: 2px; margin: 2px; width: 150px; height: 20px; border-radius: 5px; outline: none;" type="text" value="1"/>	

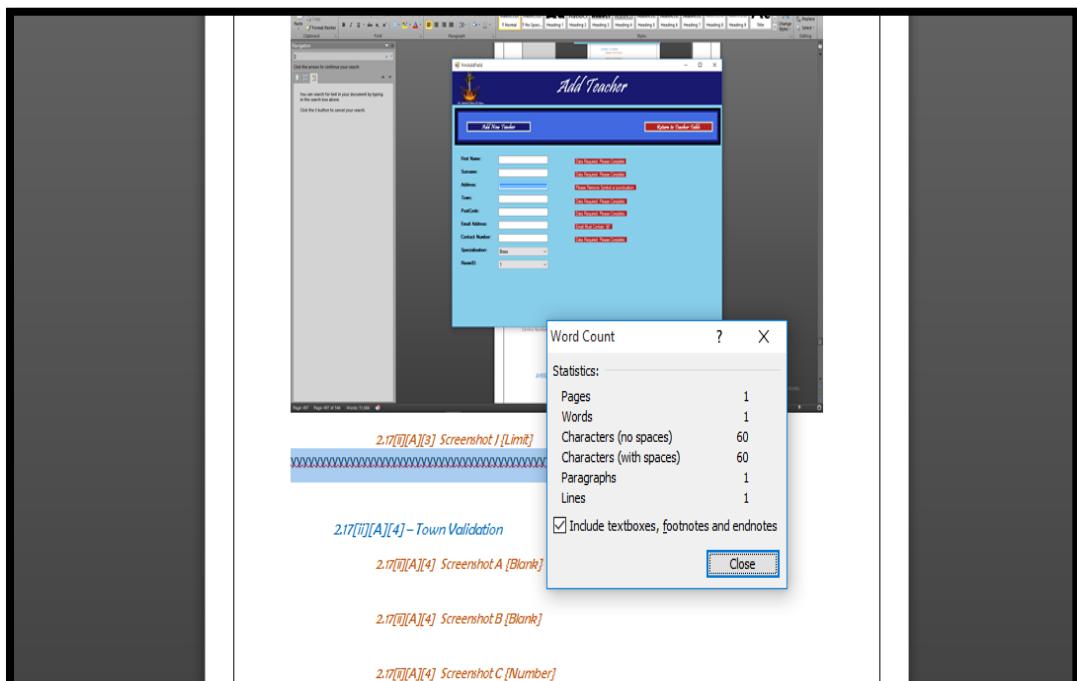
2.17[ii][A][3] Screenshot I { Limit }

frmAddField

Add Teacher

Add New Teacher *Return to Teacher Table*

First Name:	<input type="text"/>	Data Required. Please Complete.
Surname:	<input type="text"/>	Data Required. Please Complete.
Address:	<input style="border: 2px solid red; color: red; background-color: #f0f0f0; font-style: italic; font-weight: bold; font-size: 1em; padding: 2px; margin: 2px; width: 150px; height: 20px; border-radius: 5px; outline: none;" type="text" value="oooooooooooooo"/>	Please Remove Symbol or punctuation.
Town:	<input type="text"/>	Data Required. Please Complete.
PostCode:	<input type="text"/>	Data Required. Please Complete.
Email Address:	<input type="text"/>	Email Must Contain '@'.
Contact Number:	<input type="text"/>	Data Required. Please Complete.
Specialisation:	<input style="border: 1px solid black; color: black; background-color: white; font-style: italic; font-weight: bold; font-size: 1em; padding: 2px; margin: 2px; width: 150px; height: 20px; border-radius: 5px; outline: none;" type="text" value="Brass"/>	
RoomID:	<input style="border: 1px solid black; color: black; background-color: white; font-style: italic; font-weight: bold; font-size: 1em; padding: 2px; margin: 2px; width: 150px; height: 20px; border-radius: 5px; outline: none;" type="text" value="1"/>	

2.17[ii][A][3] Screenshot J {Limit}**2.17[ii][A][4] – Town Validation****2.17[ii][A][4] Screenshot A {Blank}**

frmAddField

Add Teacher

Add New Teacher **Return to Teacher Table**

First Name:	<input type="text"/>	Data Required. Please Complete.
Surname:	<input type="text"/>	Data Required. Please Complete.
Address:	<input type="text"/>	Data Required. Please Complete.
Town:	<input type="text"/>	Data Required. Please Complete.
PostCode:	<input type="text"/>	Data Required. Please Complete.
Email Address:	<input type="text"/>	Email Must Contain "@".
Contact Number:	<input type="text"/>	Data Required. Please Complete.
Specialisation:	Brass	
RoomID:	1	

2.17[ii][A][4] Screenshot B {Blank}

The screenshot shows a Windows application window titled "frmAddField". The title bar features a logo of a guitar and the text "The Mitchell School Of Music". The main title of the window is "Add Teacher". Below the title, there are two buttons: "Add New Teacher" (in blue) and "Return to Teacher Table" (in red). The form contains nine input fields with validation messages:

Field	Value	Validation Message
First Name:		Data Required. Please Complete.
Surname:		Data Required. Please Complete.
Address:		Data Required. Please Complete.
Town:	test	
PostCode:		Data Required. Please Complete.
Email Address:		Email Must Contain '@'.
Contact Number:		Data Required. Please Complete.
Specialisation:	Brass	
RoomID:	1	

2.17[ii][A][4] Screenshot C {Number}

The screenshot shows the same Windows application window "frmAddField" with the "Add Teacher" title. The validation messages have changed:

Field	Value	Validation Message
First Name:		Data Required. Please Complete.
Surname:		Data Required. Please Complete.
Address:		Data Required. Please Complete.
Town:	test2	Please remove numerical characters.
PostCode:		Data Required. Please Complete.
Email Address:		Email Must Contain '@'.
Contact Number:		Data Required. Please Complete.
Specialisation:	Brass	
RoomID:	1	

2.17[ii][A][4] Screenshot D {Symbol}

The screenshot shows the 'Add Teacher' form with several validation errors:

Field	Error Message
First Name:	Data Required. Please Complete.
Surname:	Data Required. Please Complete.
Address:	Data Required. Please Complete.
Town:	Please remove symbols and punctuation.
PostCode:	Data Required. Please Complete.
Email Address:	Email Must Contain '@'.
Contact Number:	Data Required. Please Complete.
Specialisation:	Brass
RoomID:	1

2.17[ii][A][4] Screenshot E {Punctuation}

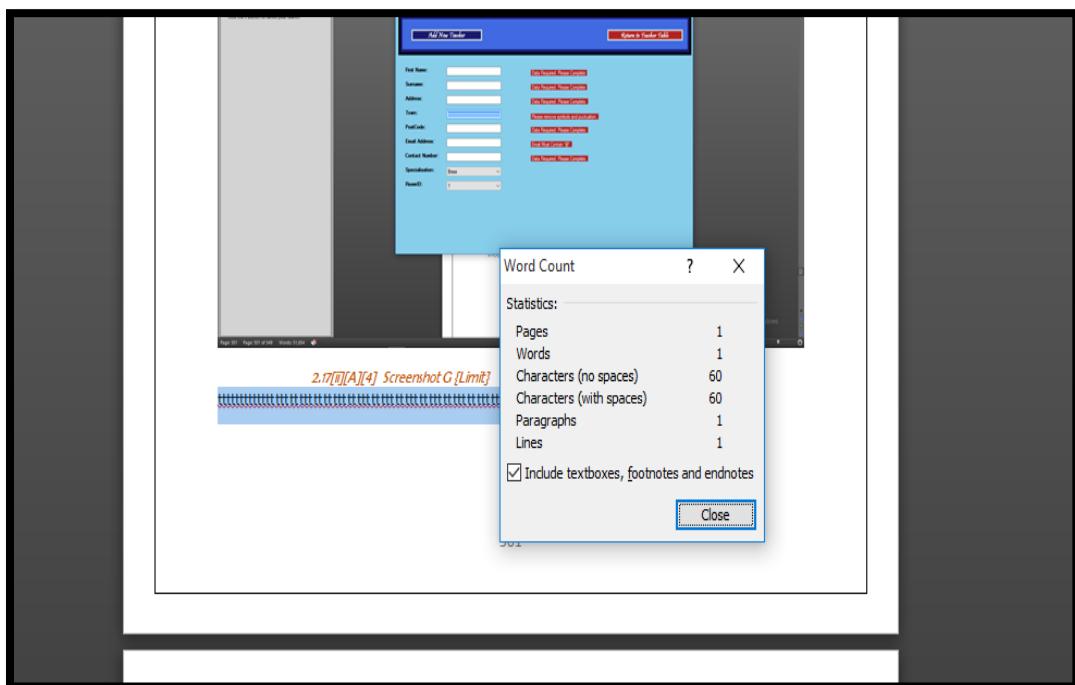
The screenshot shows the 'Add Teacher' form with several validation errors:

Field	Error Message
First Name:	Data Required. Please Complete.
Surname:	Data Required. Please Complete.
Address:	Data Required. Please Complete.
Town:	Please remove symbols and punctuation.
PostCode:	Data Required. Please Complete.
Email Address:	Email Must Contain '@'.
Contact Number:	Data Required. Please Complete.
Specialisation:	Brass
RoomID:	1

2.17[ii][A][4] Screenshot F {Limit}

The screenshot shows a Windows application window titled "frmAddField". The main title bar says "Add Teacher". Below it is a sub-header "Add New Teacher" and a "Return to Teacher Table" button. The form contains fields for First Name, Surname, Address, Town, PostCode, Email Address, Contact Number, Specialisation (with a dropdown menu showing "Brass"), and RoomID (with a dropdown menu showing "1"). Most fields have red validation error messages to their right: "Data Required. Please Complete." for First Name, Surname, and Address; "Please remove symbols and punctuation." for Town; "Data Required. Please Complete." for PostCode; "Email Must Contain '@'." for Email Address; and "Data Required. Please Complete." for Contact Number.

2.17[ii][A][4] Screenshot G {Limit}



2.17[ii][A][5] – Post Code Validation**2.17[ii][A][5] Screenshot A {Blank}**

The screenshot shows a Windows application window titled "frmAddField". The main title bar says "Add Teacher". Below the title bar is a logo of a guitar and the text "The Mitchell School Of Music". The interface has a blue header bar with two buttons: "Add New Teacher" and "Return to Teacher Table". The main area contains nine input fields with validation messages:

Field	Value	Validation Message
First Name:		Data Required. Please Complete.
Surname:		Data Required. Please Complete.
Address:		Data Required. Please Complete.
Town:		Data Required. Please Complete.
PostCode:		Data Required. Please Complete.
Email Address:		Email Must Contain '@'.
Contact Number:		Data Required. Please Complete.
Specialisation:	Brass	
RoomID:	1	

2.17[ii][A][5] Screenshot B {Blank}

The screenshot shows the same Windows application window as in Screenshot A. The main title bar says "Add Teacher". Below the title bar is the logo of a guitar and the text "The Mitchell School Of Music". The interface has a blue header bar with two buttons: "Add New Teacher" and "Return to Teacher Table". The main area contains the same nine input fields as in Screenshot A, but with different values and validation messages:

Field	Value	Validation Message
First Name:		Data Required. Please Complete.
Surname:		Data Required. Please Complete.
Address:		Data Required. Please Complete.
Town:		Data Required. Please Complete.
PostCode:	BT423EW	
Email Address:		Email Must Contain '@'.
Contact Number:		Data Required. Please Complete.
Specialisation:	Brass	
RoomID:	1	

2.17[ii][A][5] Screenshot C {BT}

The screenshot shows the 'Add Teacher' form from the 'frmAddField' application. The window title is 'frmAddField' and the main title is 'Add Teacher'. The form contains fields for First Name, Surname, Address, Town, PostCode, Email Address, Contact Number, Specialisation, and RoomID. Most fields have validation errors displayed next to them in red text.

Field	Value	Error Message
First Name		Data Required. Please Complete.
Surname		Data Required. Please Complete.
Address		Data Required. Please Complete.
Town		Data Required. Please Complete.
PostCode	BE423EW	[PostCode Format = LL000LL] First characters must be 'BT'.
Email Address		Email Must Contain '@'.
Contact Number		Data Required. Please Complete.
Specialisation	Brass	
RoomID	1	

2.17[ii][A][5] Screenshot D {Letters – Number?}

The screenshot shows the 'Add Teacher' form from the 'frmAddField' application. The window title is 'frmAddField' and the main title is 'Add Teacher'. The form contains fields for First Name, Surname, Address, Town, PostCode, Email Address, Contact Number, Specialisation, and RoomID. Most fields have validation errors displayed next to them in red text.

Field	Value	Error Message
First Name		Data Required. Please Complete.
Surname		Data Required. Please Complete.
Address		Data Required. Please Complete.
Town		Data Required. Please Complete.
PostCode	BT4232W	[PostCode Format = LL000LL] Please Replace final digits with valid characters.
Email Address		Email Must Contain '@'.
Contact Number		Data Required. Please Complete.
Specialisation	Brass	
RoomID	1	

2.17[ii][A][5] Screenshot E {Letters – Symbol}

The screenshot shows a Windows application window titled "frmAddField". The main title bar says "Add Teacher". Below the title bar is a logo of a guitar and the text "The Mitchell School Of Music". The interface has a blue header bar with two buttons: "Add New Teacher" and "Return to Teacher Table". The main area contains form fields for entering teacher information. Most fields have validation errors displayed next to them in red text:

Field	Error Message
First Name:	Data Required. Please Complete.
Surname:	Data Required. Please Complete.
Address:	Data Required. Please Complete.
Town:	Data Required. Please Complete.
PostCode:	[PostCode Format = LL000LL] Please Replace final digits with valid characters..
Email Address:	Email Must Contain '@'.
Contact Number:	Data Required. Please Complete.
Specialisation:	Brass
RoomID:	1

2.17[ii][A][5] Screenshot F {Letters – Punctuation}

The screenshot shows the same Windows application window as in Screenshot E, titled "frmAddField" with "Add Teacher" in the title bar. The logo and buttons are identical. The form fields and their validation errors are as follows:

Field	Error Message
First Name:	Data Required. Please Complete.
Surname:	Data Required. Please Complete.
Address:	Data Required. Please Complete.
Town:	Data Required. Please Complete.
PostCode:	[PostCode Format = LL000LL] Please Replace final digits with valid characters..
Email Address:	Email Must Contain '@'.
Contact Number:	Data Required. Please Complete.
Specialisation:	Brass
RoomID:	1

2.17[ii][A][5] Screenshot G {Numbers – Letter}

frmAddField

Add Teacher

[Add New Teacher](#) [Return to Teacher Table](#)

First Name:	<input type="text"/>	Data Required. Please Complete.
Surname:	<input type="text"/>	Data Required. Please Complete.
Address:	<input type="text"/>	Data Required. Please Complete.
Town:	<input type="text"/>	Data Required. Please Complete.
PostCode:	<input type="text" value="BT42EER"/>	[PostCode Format = LL000LL] Please Replace '000' section with valid characters.
Email Address:	<input type="text"/>	Email Must Contain '@'.
Contact Number:	<input type="text"/>	Data Required. Please Complete.
Specialisation:	<input type="text" value="Brass"/>	
RoomID:	<input type="text" value="1"/>	

2.17[ii][A][5] Screenshot H {Limit }

frmAddField

Add Teacher

[Add New Teacher](#) [Return to Teacher Table](#)

First Name:	<input type="text"/>	Data Required. Please Complete.
Surname:	<input type="text"/>	Data Required. Please Complete.
Address:	<input type="text"/>	Data Required. Please Complete.
Town:	<input type="text"/>	Data Required. Please Complete.
PostCode:	<input type="text" value="BT435e"/>	[PostCode Format = LL000LL] Please use full format.
Email Address:	<input type="text"/>	Email Must Contain '@'.
Contact Number:	<input type="text"/>	Data Required. Please Complete.
Specialisation:	<input type="text" value="Brass"/>	
RoomID:	<input type="text" value="1"/>	

2.17[ii][A][6] – Email Address Validation**2.17[ii][A][6] Screenshot A {Blank}**

The screenshot shows a Windows application window titled "frmAddField". The main title bar says "Add Teacher". The window contains a logo of a guitar and the text "The Notched School Of Music". There are two buttons at the top: "Add New Teacher" (blue) and "Return to Teacher Table" (red). The form has the following fields and validation messages:

Field	Value	Validation Message
First Name:		Data Required. Please Complete.
Surname:		Data Required. Please Complete.
Address:		Data Required. Please Complete..
Town:		Data Required. Please Complete..
PostCode:		Data Required. Please Complete..
Email Address:		Email Must Contain '@'.
Contact Number:		Data Required. Please Complete..
Specialisation:	Brass	
RoomID:	1	

2.17[ii][A][6] Screenshot B {Blank}

The screenshot shows the same Windows application window as Screenshot A. The validation message for the Email Address field has changed to "Email Must Contain '@'.". The other validation messages remain the same as in Screenshot A.

Field	Value	Validation Message
First Name:		Data Required. Please Complete.
Surname:		Data Required. Please Complete.
Address:		Data Required. Please Complete..
Town:		Data Required. Please Complete..
PostCode:		Data Required. Please Complete..
Email Address:	Test@Test.com	Email Must Contain '@'.
Contact Number:		Data Required. Please Complete..
Specialisation:	Brass	
RoomID:	1	

2.17[ii][A][6] Screenshot C {No "@"}

The screenshot shows the 'Add Teacher' form from 'The National School Of Music'. The form has a blue header bar with 'Add New Teacher' and 'Return to Teacher Table' buttons. Below the header are nine input fields for First Name, Surname, Address, Town, PostCode, Email Address, Contact Number, Specialisation, and RoomID. Each field has a red validation message to its right. The validation messages are: 'Data Required. Please Complete.' for First Name, Surname, Address, Town, PostCode, and Contact Number; 'Email Must Contain "@".' for Email Address; and 'Data Required. Please Complete.' for Specialisation.

Field	Value	Validation Message
First Name		Data Required. Please Complete.
Surname		Data Required. Please Complete.
Address		Data Required. Please Complete.
Town		Data Required. Please Complete.
PostCode		Data Required. Please Complete.
Email Address	Test	Email Must Contain "@".
Contact Number		Data Required. Please Complete.
Specialisation	Brass	Data Required. Please Complete.
RoomID	1	

2.17[ii][A][6] Screenshot D {Multiple "@"s}

The screenshot shows the 'Add Teacher' form from 'The National School Of Music'. The form has a blue header bar with 'Add New Teacher' and 'Return to Teacher Table' buttons. Below the header are nine input fields for First Name, Surname, Address, Town, PostCode, Email Address, Contact Number, Specialisation, and RoomID. The Email Address field contains 'Test@Test@Test.com'. A red validation message 'Only one "@" may be present within Email.' appears next to it. All other fields are empty and show the validation message 'Data Required. Please Complete.'

Field	Value	Validation Message
First Name		Data Required. Please Complete.
Surname		Data Required. Please Complete.
Address		Data Required. Please Complete.
Town		Data Required. Please Complete.
PostCode		Data Required. Please Complete.
Email Address	Test@Test@Test.com	Only one "@" may be present within Email.
Contact Number		Data Required. Please Complete.
Specialisation	Brass	Data Required. Please Complete.
RoomID	1	

2.17[ii][A][6] Screenshot E {Limit}

Add Teacher

Add New Teacher **Return to Teacher Table**

First Name: [Text Box] Data Required. Please Complete.

Surname: [Text Box] Data Required. Please Complete.

Address: [Text Box] Data Required. Please Complete.

Town: [Text Box] Data Required. Please Complete.

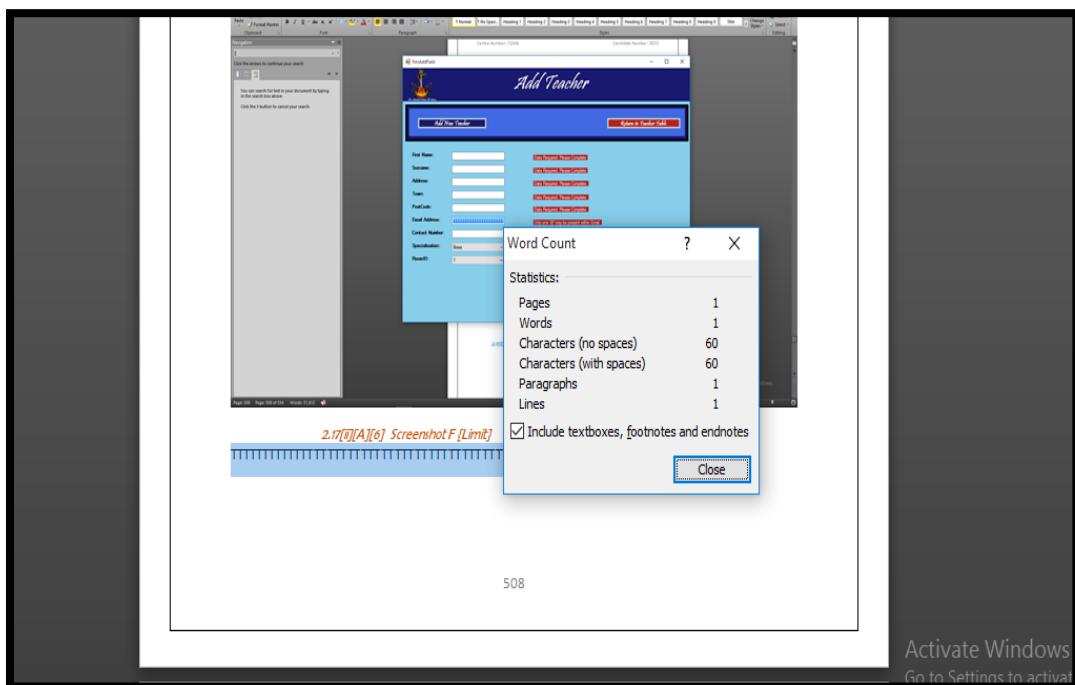
PostCode: [Text Box] Data Required. Please Complete.

Email Address: [Text Box] Only one "@" may be present within Email.

Contact Number: [Text Box] Data Required. Please Complete.

Specialisation: Brass

RoomID: 1

2.17[ii][A][6] Screenshot F {Limit}

2.17[ii][A][7] – Contact Number Validation**2.17[ii][A][7] Screenshot A {Blank}**

The screenshot shows the 'Add Teacher' form from 'The Mitchell School Of Music'. The form has a blue header with the title 'Add Teacher'. Below the header are two buttons: 'Add New Teacher' (blue background) and 'Return to Teacher Table' (red background). The main area contains eight input fields with their respective validation messages:

Field	Value	Validation Message
First Name:	[Empty]	Data Required. Please Complete.
Surname:	[Empty]	Data Required. Please Complete.
Address:	[Empty]	Data Required. Please Complete.
Town:	[Empty]	Data Required. Please Complete.
PostCode:	[Empty]	Data Required. Please Complete.
Email Address:	[Empty]	Email Must Contain '@'.
Contact Number:	[Empty]	Data Required. Please Complete.
Specialisation:	Brass	
RoomID:	1	

2.17[ii][A][7] Screenshot B {Blank}

The screenshot shows the same 'Add Teacher' form as in Screenshot A. The validation message for the 'Contact Number' field has changed to '02784484848'. All other fields still show validation errors: First Name, Surname, Address, Town, PostCode, Email Address, and RoomID.

Field	Value	Validation Message
First Name:	[Empty]	Data Required. Please Complete.
Surname:	[Empty]	Data Required. Please Complete.
Address:	[Empty]	Data Required. Please Complete.
Town:	[Empty]	Data Required. Please Complete.
PostCode:	[Empty]	Data Required. Please Complete.
Email Address:	[Empty]	Email Must Contain '@'.
Contact Number:	02784484848	
Specialisation:	Brass	
RoomID:	1	

2.17[ii][A][7] Screenshot C {Letter Present}

The screenshot shows the 'Add Teacher' form from the frmAddField application. The window title is 'frmAddField' and the main title is 'Add Teacher'. The form contains fields for First Name, Surname, Address, Town, PostCode, Email Address, Contact Number, Specialisation, and RoomID. Most fields have validation errors displayed next to them in red boxes:

- First Name: Data Required. Please Complete.
- Surname: Data Required. Please Complete.
- Address: Data Required. Please Complete.
- Town: Data Required. Please Complete.
- PostCode: Data Required. Please Complete.
- Email Address: Email Must Contain '@'.
- Contact Number: Please Remove Alphabetical Characters.

The 'Specialisation' dropdown is set to 'Brass' and the 'RoomID' dropdown is set to '1'.

2.17[ii][A][7] Screenshot D {Symbol Present}

The screenshot shows the 'Add Teacher' form from the frmAddField application. The window title is 'frmAddField' and the main title is 'Add Teacher'. The form contains fields for First Name, Surname, Address, Town, PostCode, Email Address, Contact Number, Specialisation, and RoomID. Most fields have validation errors displayed next to them in red boxes:

- First Name: Data Required. Please Complete.
- Surname: Data Required. Please Complete.
- Address: Data Required. Please Complete.
- Town: Data Required. Please Complete.
- PostCode: Data Required. Please Complete.
- Email Address: Email Must Contain '@'.
- Contact Number: Please Remove symbols.

The 'Specialisation' dropdown is set to 'Brass' and the 'RoomID' dropdown is set to '1'. The 'Contact Number' field contains the value '0278448484\$'.

2.17[ii][A][7] Screenshot E {Punctuation Present}

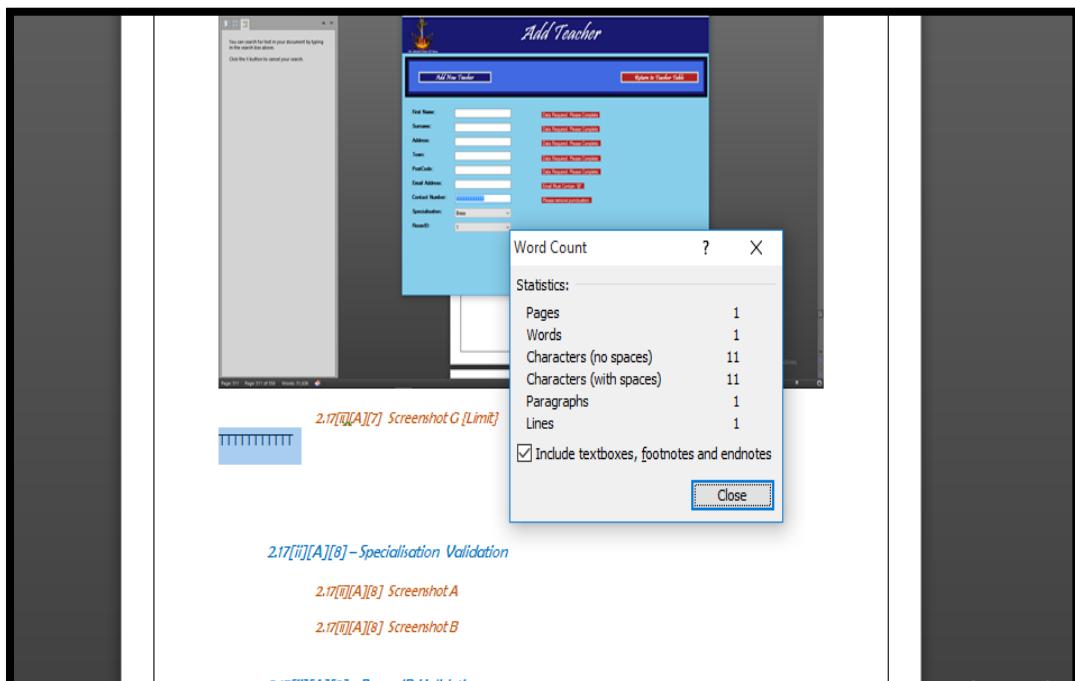
The screenshot shows the frmAddField application window titled "Add Teacher". The form contains the following fields and their validation messages:

Field	Value	Validation Message
First Name:		Data Required. Please Complete.
Surname:		Data Required. Please Complete.
Address:		Data Required. Please Complete.
Town:		Data Required. Please Complete.
PostCode:		Data Required. Please Complete.
Email Address:		Email Must Contain '@'.
Contact Number:	0278448484"	Please remove punctuation.
Specialisation:	Brass	
RoomID:	1	

2.17[ii][A][7] Screenshot F {Limit}

The screenshot shows the frmAddField application window titled "Add Teacher". The form contains the following fields and their validation messages:

Field	Value	Validation Message
First Name:		Data Required. Please Complete.
Surname:		Data Required. Please Complete.
Address:		Data Required. Please Complete.
Town:		Data Required. Please Complete.
PostCode:		Data Required. Please Complete.
Email Address:		Email Must Contain '@'.
Contact Number:	0278448484	Please remove punctuation.
Specialisation:	Brass	
RoomID:	1	

2.17[ii][A][7] Screenshot G {Limit}**2.17[ii][A][8] – Specialisation Validation****2.17[ii][A][8] Screenshot A**

The screenshot shows the 'Add Teacher' form with several validation errors. The validation messages are displayed in red boxes next to the corresponding input fields:

- First Name:** Data Required. Please Complete
- Surname:** Data Required. Please Complete
- Address:** Data Required. Please Complete
- Town:** Data Required. Please Complete
- PostCode:** Data Required. Please Complete
- Email Address:** Email Must Contain '@'.
- Contact Number:** Data Required. Please Complete
- Specialisation:** Brass
- RoomID:** Brass
Keyboard
Percussion
String
Vocal
WoodWind

2.17[ii][A][8] Screenshot B

The screenshot shows the 'Add Teacher' form with the following validation errors:

- First Name:** Data Required. Please Complete.
- Surname:** Data Required. Please Complete.
- Address:** Data Required. Please Complete.
- Town:** Data Required. Please Complete.
- PostCode:** Data Required. Please Complete.
- Email Address:** Email Must Contain '@'.
- Contact Number:** Data Required. Please Complete.
- Specialisation:** Vocal
- RoomID:** 1

2.17[ii][A][9] – Room ID Validation**2.17[iii][A][9] Screenshot A**

The screenshot shows the 'Add Teacher' form with the following validation errors:

- First Name:** Data Required. Please Complete.
- Surname:** Data Required. Please Complete.
- Address:** Data Required. Please Complete.
- Town:** Data Required. Please Complete.
- PostCode:** Data Required. Please Complete.
- Email Address:** Data Required. Please Complete.
- Contact Number:** Data Required. Please Complete.
- Specialisation:** Vocal
- RoomID:** RoomID dropdown menu showing values from 1 to 17, with value 13 selected.

A tooltip at the bottom of the RoomID dropdown says: "2.17[iii][B][1] – Teacher Information Display".

2.17[ii][A][9] Screenshot B

Add Teacher

Add New Teacher **Return to Teacher Table**

First Name: [Text Box] Data Required. Please Complete.

Surname: [Text Box] Data Required. Please Complete.

Address: [Text Box] Data Required. Please Complete.

Town: [Text Box] Data Required. Please Complete.

PostCode: [Text Box] Data Required. Please Complete.

Email Address: [Text Box] Email Must Contain '@'.

Contact Number: [Text Box] Data Required. Please Complete.

Specialisation: Vocal

RoomID: 13

Process [B]) Update Display**2.17[ii][B][1] – Teacher Information Display****2.17[ii][B][1] Screenshot A**

Teachers List

TeacherID: 1

First Name: Sandra

Surname: Smith

Address: 362 Atlantic Road

Town: Coleraine

Post Code: BT32 3RE

Contact Number: 84933740

Email Address: ssmith362@gmail.co.uk

Specialisation: String

RoomID: 1

Search for Teacher: Sandra

previous Teacher Record **Next Teacher Record**

Add new Teacher

Update Teacher Record

Remove Teacher Record

Return to Main Menu

2.17[ii][B][1] Screenshot B

The screenshot shows a Windows application window titled "frmAddField". The main title bar says "Add Teacher". Below the title bar is a logo of a guitar and the text "The Mitchell School Of Music". The window has a blue header bar with "Add New Teacher" and "Return to Teacher Table" buttons. The main area contains form fields for teacher details:

Teacher ID	1
First Name:	Sandra
Surname:	Smith
Address:	362 Atlantic Road
Town:	Coleraine
PostCode:	BT323RE
Email Address:	ssmith362@gmail.co.uk
Contact Number:	84933740
Specialisation:	Brass
RoomID:	1

2.17[ii] Instrument Record**Process [A]) Validation****2.17[iii][A][1] – Instrument Type Validation****2.17[ii][A][1] Screenshot A**

The screenshot shows a Windows application window titled "frmAddField". The main title bar says "Add Instrument". Below the title bar is a logo of a guitar and the text "The Mitchell School Of Music". The window has a blue header bar with "Add New Instrument" and "Return to Instrument Table" buttons. The main area contains form fields for instrument details:

Instrument Type:	Brass
Instrument Name:	Keyboard
Quantity:	Percussion

A dropdown menu is open under "Instrument Type", showing options: Brass, Keyboard, Percussion, String, Vocal, and WoodWind. The option "String" is currently selected.

2.17[ii][A][1] Screenshot B

The screenshot shows a Windows application window titled "frmAddField". The main title bar has a logo of a guitar and the text "The Mitchell School Of Music". The main title of the window is "Add Instrument". Below the title, there are two buttons: "Add New Instrument" on the left and "Return to Instrument Table" on the right. The main area contains three input fields: "Instrument Type" with a dropdown menu set to "Vocal", "Instrument Name" with an empty text box, and "Quantity" with an empty text box.

2.17[iii][A][2] – Instrument Name Validation**2.17[ii][A][2] Screenshot A {Blank}**

This screenshot shows the same "frmAddField" window as above, but with validation errors. The "Instrument Name" field is empty, and the "Quantity" field is also empty. Red validation messages appear next to each field: "Data Required. Please Complete." for both.

2.17[ii][A][2] Screenshot B {Blank}

The screenshot shows a Windows application window titled "frmAddField". The title bar features a logo of a guitar on fire and the text "The Mitchell School Of Music". The main title of the window is "Add Instrument". A blue header bar contains two buttons: "Add New Instrument" on the left and "Return to Instrument Table" on the right. The main body of the window has three input fields: "Instrument Type" with a dropdown menu set to "Vocal", "Instrument Name" with the value "Test", and "Quantity" with an empty text box. A red error message "Data Required. Please Complete." is displayed next to the empty quantity field.

2.17[ii][A][2] Screenshot C {Number Present}

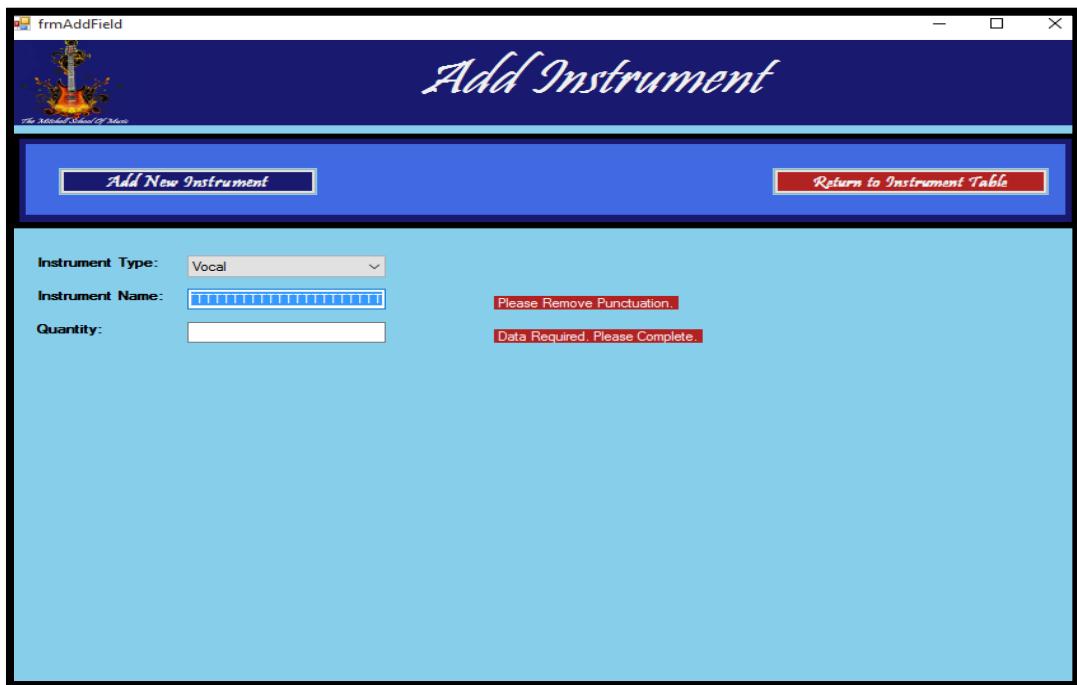
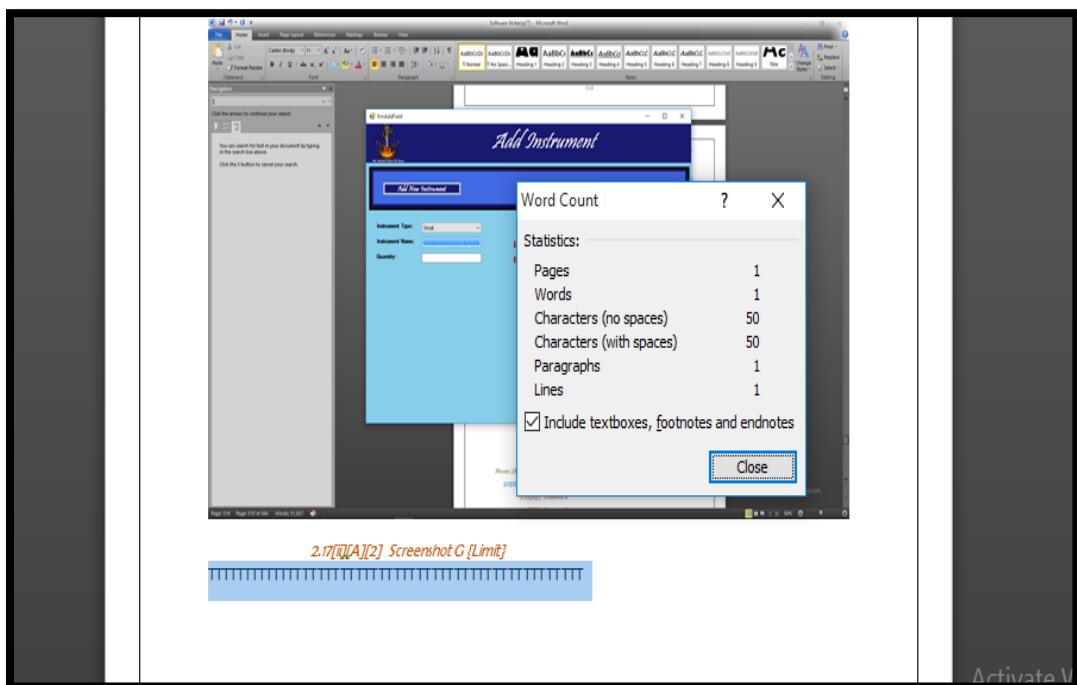
The screenshot shows the same Windows application window as in Screenshot B. The "Instrument Name" field now contains "Test2", which includes a numeric character. A red error message "Please Remove Numeric Character." is shown next to the instrument name field. The "Quantity" field remains empty, and the red error message "Data Required. Please Complete." is still present next to it.

2.17[ii][A][2] Screenshot D {Symbol Present}

The screenshot shows the 'frmAddField' application window titled 'Add Instrument'. The interface includes a logo for 'The Musical School Of Music' featuring a stylized guitar. At the top, there are two buttons: 'Add New Instrument' (blue background) and 'Return to Instrument Table' (red background). The main form contains three input fields: 'Instrument Type' (dropdown menu set to 'Vocal'), 'Instrument Name' (text input containing 'Test£'), and 'Quantity' (text input). Error messages are displayed next to the name and quantity fields: 'Please Remove Symbol Character.' and 'Data Required. Please Complete.' respectively.

2.17[ii][A][2] Screenshot E {Punctuation Present}

The screenshot shows the 'frmAddField' application window titled 'Add Instrument'. The interface includes a logo for 'The Musical School Of Music' featuring a stylized guitar. At the top, there are two buttons: 'Add New Instrument' (blue background) and 'Return to Instrument Table' (red background). The main form contains three input fields: 'Instrument Type' (dropdown menu set to 'Vocal'), 'Instrument Name' (text input containing 'Test"'), and 'Quantity' (text input). Error messages are displayed next to the name and quantity fields: 'Please Remove Punctuation.' and 'Data Required. Please Complete.' respectively.

2.17[ii][A][2] Screenshot F {Limit}**2.17[ii][A][2] Screenshot G {Limit}**

2.17[iii][A][3] – Quantity Validation**2.17[ii][A][3] Screenshot A {Blank}**

The screenshot shows the 'Add Instrument' form from 'The Mitchell School Of Music'. The 'Instrument Type' dropdown is set to 'Vocal'. The 'Instrument Name' field is empty and highlighted in red, with the error message 'Data Required. Please Complete.' displayed next to it. The 'Quantity' field is also empty and highlighted in red, with the same error message displayed next to it. The 'Add New Instrument' button is on the left, and the 'Return to Instrument Table' button is on the right.

2.17[ii][A][3] Screenshot B {Blank}

The screenshot shows the 'Add Instrument' form from 'The Mitchell School Of Music'. The 'Instrument Type' dropdown is set to 'Vocal'. The 'Instrument Name' field is empty and highlighted in red, with the error message 'Data Required. Please Complete.' displayed next to it. The 'Quantity' field contains the value '10' and is no longer highlighted in red. The 'Add New Instrument' button is on the left, and the 'Return to Instrument Table' button is on the right.

2.17[ii][A][3] Screenshot C {Letter Present}

The screenshot shows the 'Add Instrument' window from the 'The Musical School Of Music' application. The title bar says 'frmAddField'. The main area has a blue header with 'Add New Instrument' and 'Return to Instrument Table' buttons. Below the header, there are three input fields: 'Instrument Type' (Vocal), 'Instrument Name' (empty), and 'Quantity' (ee). Error messages are displayed next to the 'Instrument Name' and 'Quantity' fields: 'Data Required. Please Complete.' and 'Please Remove Alphabetical Characters.' respectively.

2.17[ii][A][3] Screenshot D {Symbol Present}

The screenshot shows the 'Add Instrument' window from the 'The Musical School Of Music' application. The title bar says 'frmAddField'. The main area has a blue header with 'Add New Instrument' and 'Return to Instrument Table' buttons. Below the header, there are three input fields: 'Instrument Type' (Vocal), 'Instrument Name' (empty), and 'Quantity' (\$\$). Error messages are displayed next to the 'Instrument Name' and 'Quantity' fields: 'Data Required. Please Complete.' and 'Please Remove Symbol Character.' respectively.

2.17[ii][A][3] Screenshot E {Punctuation Present}

The screenshot shows a Windows application window titled "frmAddField". The title bar features a logo of a guitar and the text "The Mitchell School Of Music". The main title of the window is "Add Instrument". A blue header bar contains two buttons: "Add New Instrument" and "Return to Instrument Table". The main body of the form has three input fields: "Instrument Type" (dropdown menu set to "Vocal"), "Instrument Name" (empty text box), and "Quantity" (text box containing "!!!"). Red validation messages are displayed next to the empty "Instrument Name" field ("Data Required. Please Complete.") and the invalid "Quantity" value ("Please Remove Punctuation.").

2.17[ii][A][3] Screenshot F {Range}

The screenshot shows the same Windows application window "frmAddField" as in Screenshot E. The "Instrument Name" field now contains the value "70". A red validation message appears below the "Quantity" field stating, "Value Exceeds limit. Please stay within the range of (1 - 20)". All other fields and the overall layout remain identical to Screenshot E.

*Process [B]) Update Display***2.17[iii][B][1] – Instrument Information Display****2.17[ii][B][1] Screenshot A**

The screenshot shows a Windows application window titled "frmInstrument". The title bar has standard minimize, maximize, and close buttons. The main area is titled "Instruments List" with a decorative graphic of a guitar and the text "The Musical School Of Music". On the left, there is a vertical stack of input fields: "Instrument ID" (containing "1"), "Instrument Type" (containing "String"), "Instrument Name" (containing "Violin"), and "Quantity" (containing "4"). To the right of these fields is a search bar with the placeholder "Search for Instrument: Violin" and arrows for navigating between records. Below the search bar are three buttons: "Add new Instrument" (green), "Update Bundle Record" (orange), and "Remove Instrument Record" (red). At the bottom is a "Return to Main Menu" button.

2.17[ii][B][1] Screenshot B

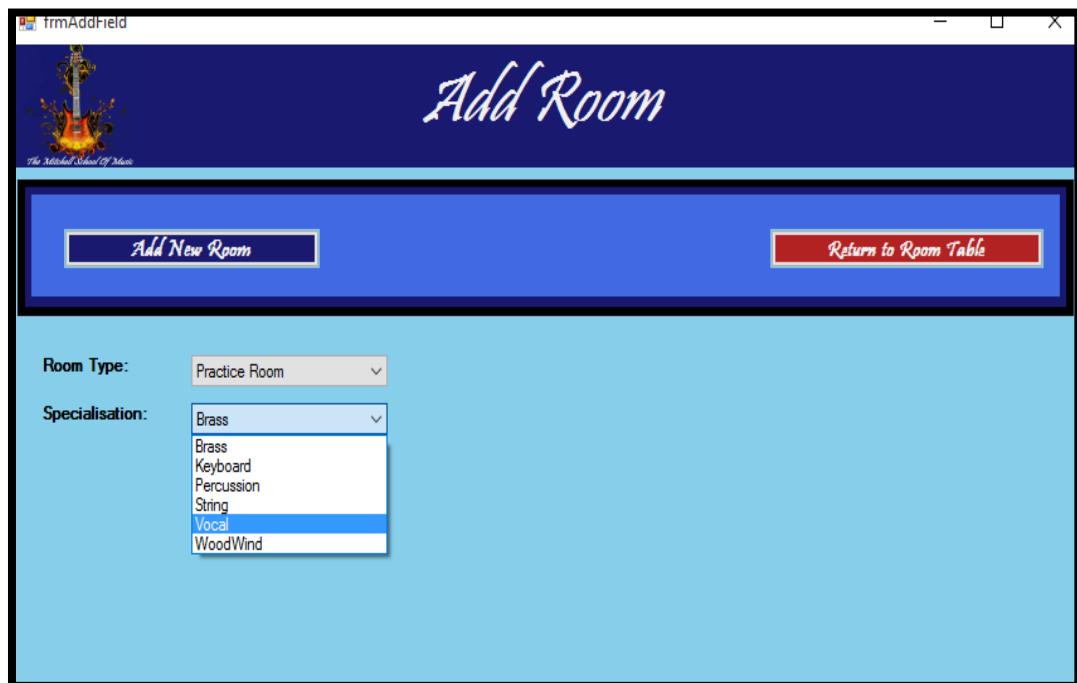
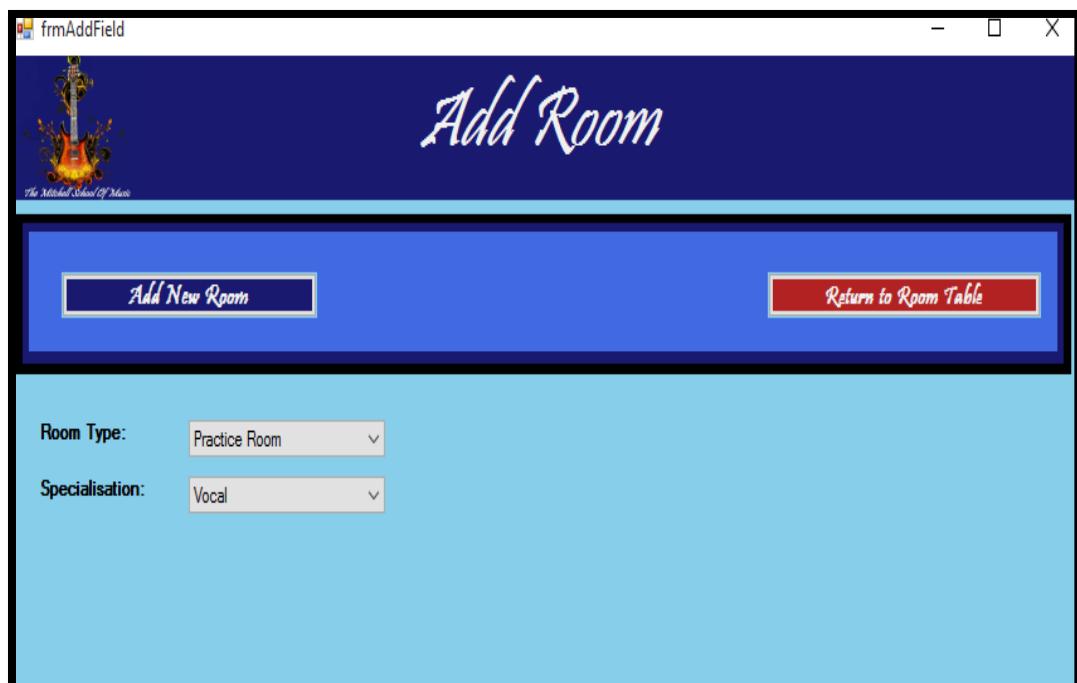
The screenshot shows a Windows application window titled "frmAddField". The title bar has standard minimize, maximize, and close buttons. The main area is titled "Add Instrument" with a decorative graphic of a guitar and the text "The Musical School Of Music". At the top are two buttons: "Add New Instrument" (left) and "Return to Instrument Table" (right). Below these buttons is a vertical stack of input fields: "Instrument ID" (containing "1"), "Instrument Type" (containing "Brass" with a dropdown arrow), "Instrument Name" (containing "Violin"), and "Quantity" (containing "4").

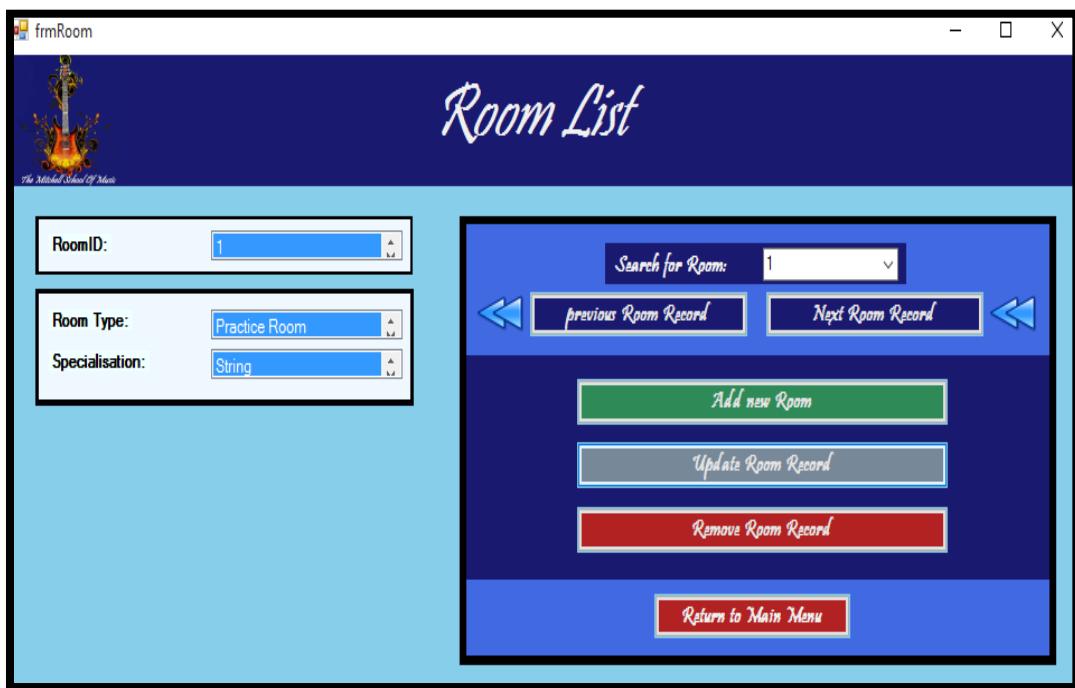
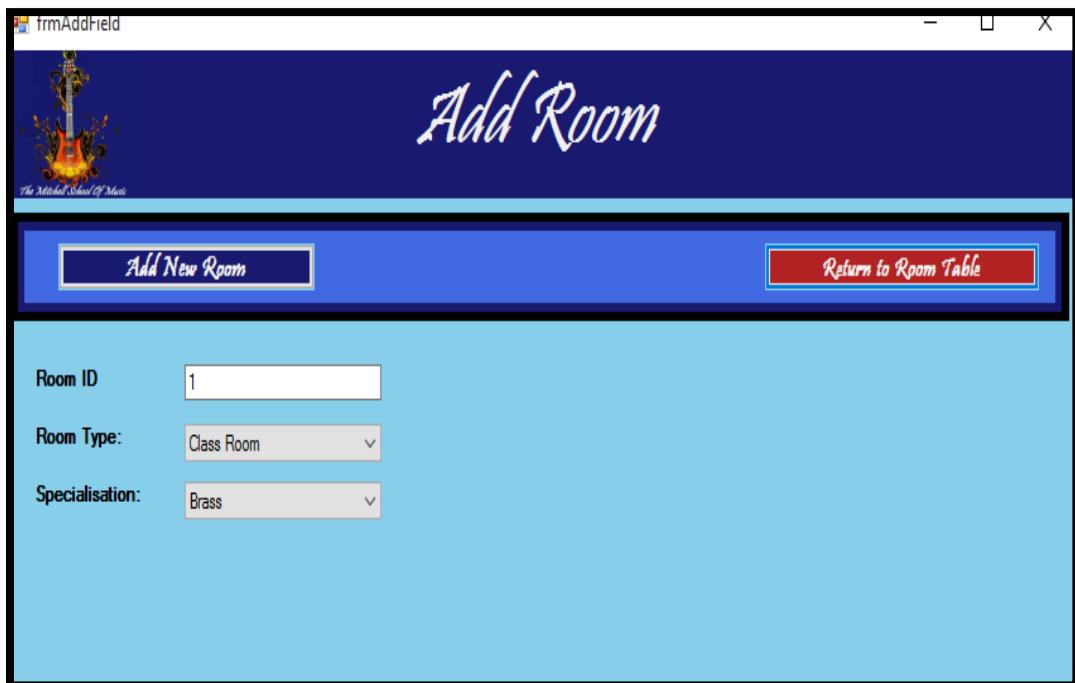
2.17[iv] Room Record**Process [A]) Validation****2.17[iv][A][i] – Room Type Validation****2.17[iv][A][i] Screenshot A**

The screenshot shows a Windows application window titled "frmAddField". The main title bar has a logo of a guitar and the text "The Mitchell School Of Music". The main title of the window is "Add Room". Below the title bar, there are two buttons: "Add New Room" on the left and "Return to Room Table" on the right. The main area contains two dropdown menus. The first dropdown is labeled "Room Type:" and has "Class Room" selected. The second dropdown is labeled "Specialisation:" and has a list of options: "Class Room", "Hall", and "Practice Room", with "Practice Room" currently highlighted.

2.17[iv][A][i] Screenshot B

The screenshot shows the same Windows application window as in Screenshot A. The main title bar and logo are identical. The main title of the window is "Add Room". Below the title bar, there are two buttons: "Add New Room" on the left and "Return to Room Table" on the right. The main area contains two dropdown menus. The first dropdown is labeled "Room Type:" and has "Practice Room" selected. The second dropdown is labeled "Specialisation:" and has "Brass" selected.

2.17[iv][A][2] – Specialisation Validation**2.17[iv][A][2] Screenshot A****2.17[iv][A][2] Screenshot B**

*Process [B]) Update Display***2.17[iv][B][1] – Room Record Display****2.17[iv][B][1] Screenshot A****2.17[iv][B][1] Screenshot B**

2.17[v] Grade Record**Process [A]) Validation****2.17[v][A][1] – Grade Title Validation****2.17[v][A][1] Screenshot A {Blank}**

The screenshot shows a Windows application window titled "frmAddField". The title bar features a logo of a guitar and the text "The Mitchell School Of Music". The main title of the window is "Add Grade". Below the title, there are two buttons: "Add New Grade" (blue border) and "Return to Grade Table" (red border). The form has two input fields: "Grade Level" and "GradeFee". Both fields have red validation messages: "Data Required. Please Complete." next to them. The "GradeLevel" field is empty, and the "GradeFee" field is also empty.

2.17[v][A][1] Screenshot B {Blank}

This screenshot is identical to Screenshot A, showing the "frmAddField" window with the "Add Grade" title. It contains the same "Add New Grade" and "Return to Grade Table" buttons. The "Grade Level" field now contains the value "Test", while the "GradeFee" field remains empty. The validation message "Data Required. Please Complete." is still displayed next to the empty "GradeFee" field.

2.17[v][A][1] Screenshot C {Number Present}

The screenshot shows a Windows application window titled "frmAddField". The title bar features a logo of a guitar and the text "The Musical School Of Music". The main title of the form is "Add Grade". On the left, there is a blue header bar with two buttons: "Add New Grade" and "Return to Grade Table". The main area contains two input fields. The first field, labeled "Grade Level:", has the value "Test3" entered. A red error message to its right says "Please Remove Numerical characters.". The second field, labeled "GradeFee:", is empty and has a red error message below it stating "Data Required. Please Complete.".

2.17[v][A][1] Screenshot D {Symbol Present}

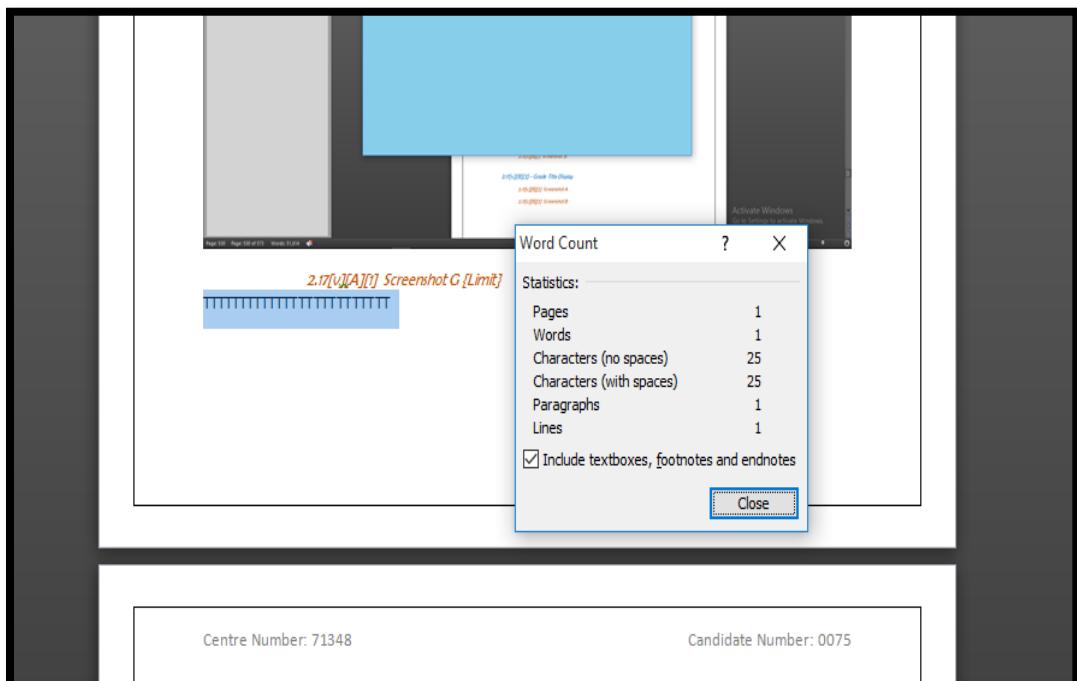
This screenshot is identical to Screenshot C, showing the "frmAddField" application window. The "Grade Level:" field still contains "Test3" with the error message "Please Remove Numerical characters.". The "GradeFee:" field remains empty with the error message "Data Required. Please Complete.".

2.17[v][A][1] Screenshot E {Punctuation Present}

The screenshot shows a Windows application window titled "frmAddField". The title bar features a logo of a guitar and the text "The Musical School Of Music". The main title of the window is "Add Grade". Below the title, there are two buttons: "Add New Grade" (blue background) and "Return to Grade Table" (red background). The form contains two text input fields. The first field, labeled "Grade Level:", contains the value "Test@". A red validation message to its right says "Please Remove Punctual characters.". The second field, labeled "GradeFee:", is empty. A red validation message to its right says "Data Required. Please Complete.". The entire application has a light blue background.

2.17[v][A][1] Screenshot F {Limit}

This screenshot shows the same Windows application window as in Screenshot E. The validation messages for the "Grade Level:" field ("Please Remove Punctual characters.") and the "GradeFee:" field ("Data Required. Please Complete.") remain the same. However, the "Grade Level:" field now contains a large number of repeated characters, such as "@@@@@@@@@", which likely exceeds a character limit set by the application's validation logic. The rest of the interface is identical to Screenshot E.

2.17[v][A][1] Screenshot G {Limit}**2.17[v][A][2] – Grade Fee Validation****2.17[v][A][2] Screenshot A {Blank}**

A screenshot of a Windows application window titled 'frmAddField'. The window has a dark blue header bar with the title 'Add Grade' and a logo for 'The Mitchell School Of Music'. Below the header is a light blue navigation bar containing two buttons: 'Add New Grade' and 'Return to Grade Table'. The main body of the window is light blue and contains two text input fields. The first field is labeled 'Grade Level:' and the second is labeled 'GradeFee:'. Both fields have red validation messages to their right: 'Data Required. Please Complete.' for each. There is also some very small, illegible text at the bottom of the window.

2.17[v][A][2] Screenshot B {Blank}

The screenshot shows a Windows application window titled "frmAddField". The title bar features a logo of a guitar and the text "The Mitchell School Of Music". The main title of the window is "Add Grade". A blue header bar contains two buttons: "Add New Grade" and "Return to Grade Table". The main body of the window has two text input fields. The first field is labeled "Grade Level:" and contains a blank white rectangle. To its right, a red error message says "Data Required. Please Complete.". The second field is labeled "GradeFee:" and contains the value "10".

2.17[v][A][2] Screenshot C {Letter Present}

This screenshot is identical to Screenshot B, but the "GradeFee:" field now contains the value "10e". A red error message to the right of the field says "Please Remove Alphabetical Characters.", indicating that the application does not accept non-numeric characters in the fee field.

2.17[v][A][2] Screenshot D {Symbol Present}

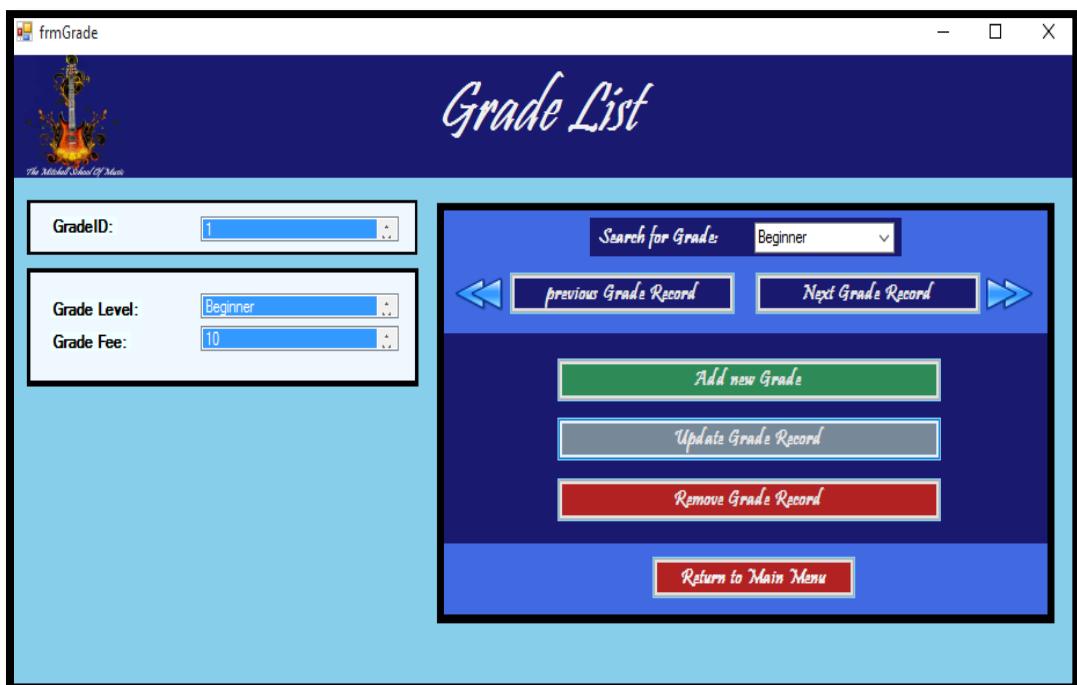
The screenshot shows a Windows application window titled "frmAddField". The title bar features a logo of a guitar and the text "The Mitchell School Of Music". The main title of the window is "Add Grade". A blue header bar contains two buttons: "Add New Grade" and "Return to Grade Table". The main area has two input fields: "Grade Level:" and "GradeFee:". The "Grade Level:" field is empty, and a red error message "Data Required. Please Complete." is displayed next to it. The "GradeFee:" field contains the value "10£", with a red error message "Please Remove Symbol Character." displayed next to it.

2.17[v][A][2] Screenshot E {Punctuation Present}

This screenshot is identical to Screenshot D, showing the "frmAddField" application window with the "Add Grade" title. The "Grade Level:" field is empty, and the error message "Data Required. Please Complete." is present. The "GradeFee:" field contains the value "10\"", and the error message "Please Remove Punctuation." is displayed next to it.

2.17[v][A][2] Screenshot F {Range}

Process [B]) Update Display

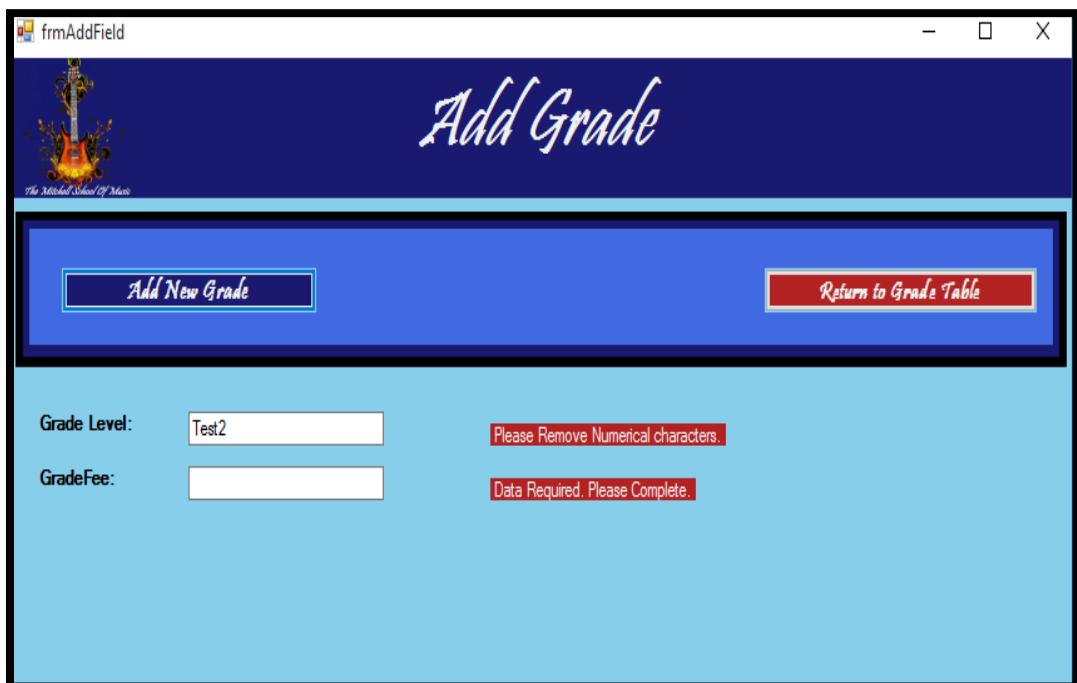
2.17[v][B][1] – Grade Record Display**2.17[v][B][1] Screenshot A**

2.17[v][B][1] Screenshot B

The screenshot shows a Windows application window titled "frmAddField". The title bar features a logo of a guitar and the text "The Mitchell School Of Music". The main title of the window is "Add Grade". On the left side, there is a decorative graphic of a guitar. Below the title, there are two buttons: "Add New Grade" and "Return to Grade Table". The central area contains three input fields: "Grade ID" with value "1", "Grade Level:" with value "Beginner", and "GradeFee:" with value "10".

2.17[vi] Lesson Bundle Record**Process [A]) Validation****2.17[vi][A][1] – Grade Title Validation****2.17[vi][A][1] Screenshot A {Blank}**

The screenshot shows the same Windows application window as before, titled "frmAddField" with the "Add Grade" title. The "Grade Level:" field is empty, and the "GradeFee:" field is also empty. To the right of each empty field, a red error message appears: "Data Required. Please Complete.".

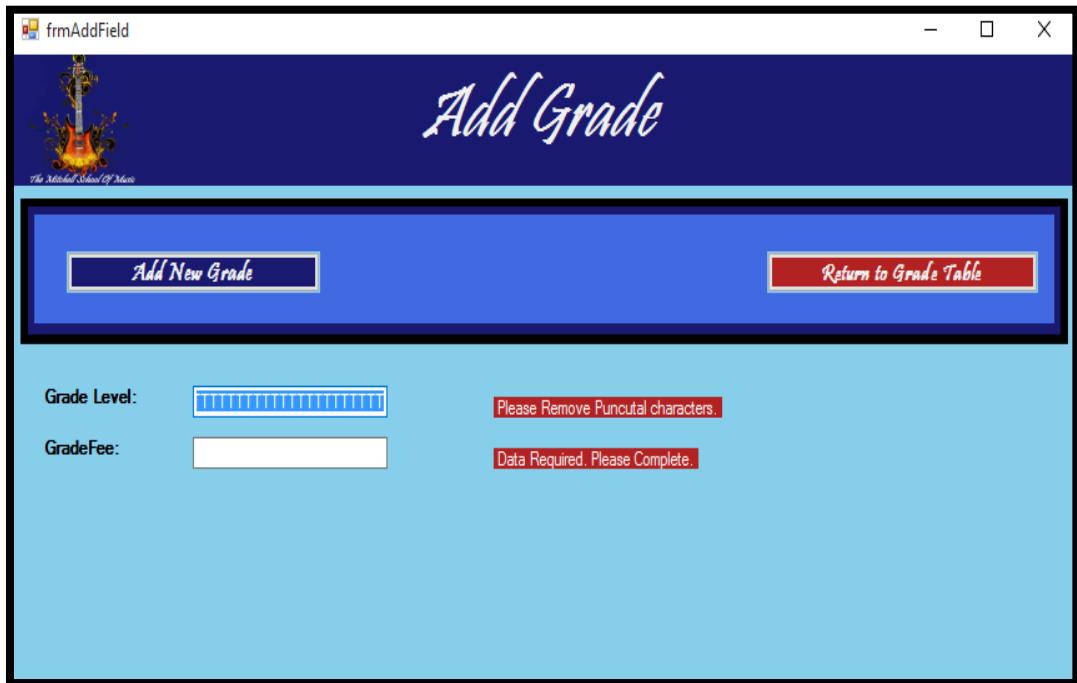
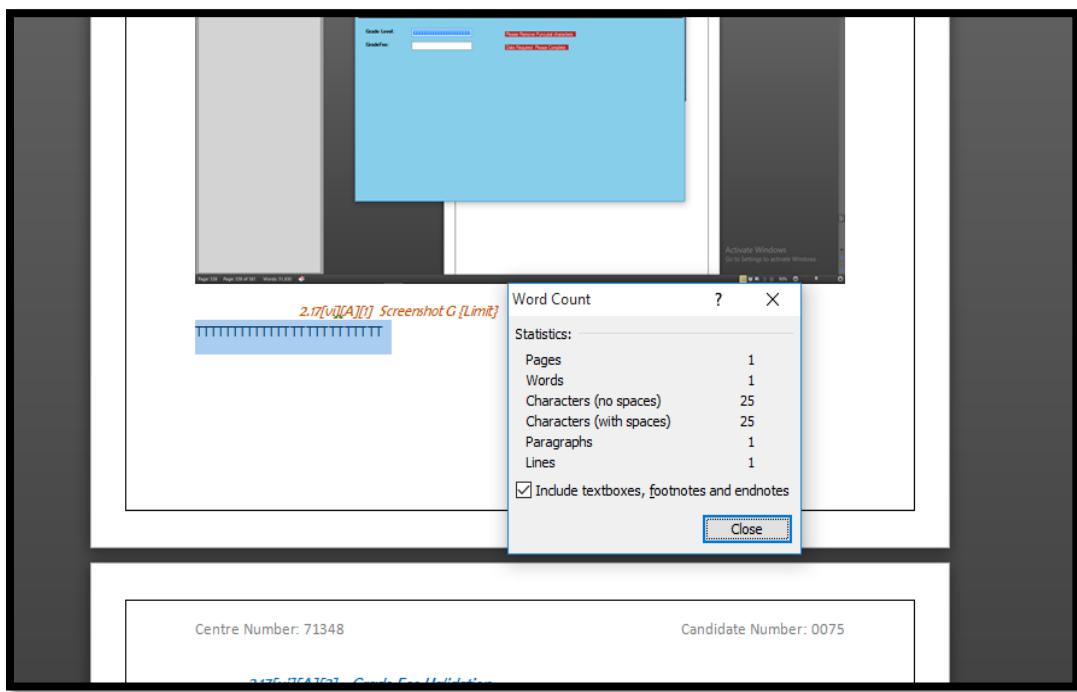
2.17[vi][A][i] Screenshot B {Blank}*2.17[vi][A][i] Screenshot C {Number Present}*

2.17[vi][A][i] Screenshot D {Symbol Present}

The screenshot shows a Windows application window titled "frmAddField". The title bar features a logo of a guitar and the text "The Mitchell School Of Music". The main title of the window is "Add Grade". Inside, there are two buttons: "Add New Grade" (blue background) and "Return to Grade Table" (red background). The "Grade Level:" field contains "Test\$". A red error message "Please Remove any Symbols." is displayed next to it. The "GradeFee:" field is empty, and a red error message "Data Required. Please Complete." is displayed next to it.

2.17[vi][A][i] Screenshot E {Punctuation Present}

This screenshot is identical to Screenshot D, showing the "frmAddField" application window. The "Grade Level:" field now contains "Test@". A red error message "Please Remove Punctual characters." is displayed next to it. The "GradeFee:" field is still empty, and a red error message "Data Required. Please Complete." is displayed next to it.

2.17[vi][A][i] Screenshot F [Limit]**2.17[vi][A][i] Screenshot G [Limit]**

2.17[vi][A][2] – Grade Fee Validation**2.17[vi][A][2] Screenshot A {Blank}**

The screenshot shows a Windows application window titled "frmAddField". The title bar features a logo of a guitar and the text "The Musical School Of Music". The main title of the window is "Add Grade". Below the title, there are two buttons: "Add New Grade" (blue background) and "Return to Grade Table" (red background). The main area contains two input fields: "Grade Level" and "GradeFee". Both fields have red validation messages: "Data Required. Please Complete." for each. The background of the main area is light blue.

2.17[vi][A][2] Screenshot B {Blank}

This screenshot is identical to Screenshot A, but the "GradeFee" field now contains the value "10". The validation message "Data Required. Please Complete." remains next to the "GradeLevel" field. The rest of the interface, including the buttons and overall layout, is the same as in Screenshot A.

2.17[vi][A][2] Screenshot C {Letter Present}

The screenshot shows a Windows application window titled "frmAddField". The title bar features a logo of a guitar and the text "The Musical School Of Music". The main title of the window is "Add Grade". A blue header bar contains two buttons: "Add New Grade" and "Return to Grade Table".
The form has two text input fields:

- "Grade Level:" followed by an empty white input field. To its right, a red error message reads "Data Required. Please Complete."
- "GradeFee:" followed by an input field containing "10e". To its right, a red error message reads "Please Remove Alphabetical Characters."

2.17[vi][A][2] Screenshot D {Symbol Present}

The screenshot shows the same Windows application window as in Screenshot C. The title bar and main title "Add Grade" are identical. The blue header bar with "Add New Grade" and "Return to Grade Table" buttons is also present.
The form has two text input fields:

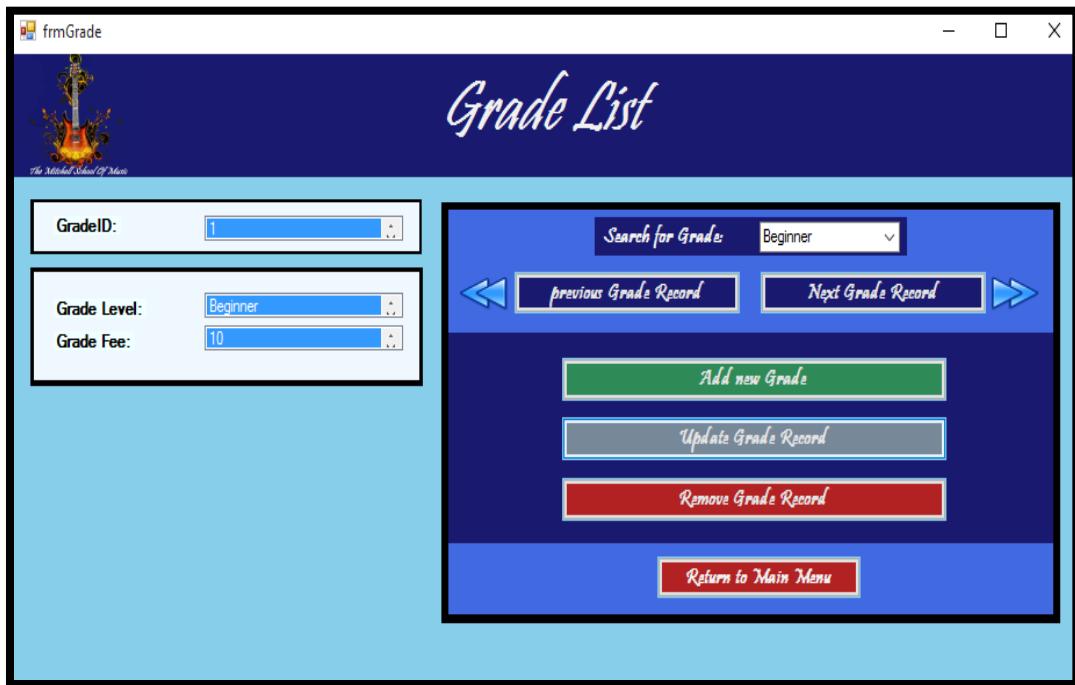
- "Grade Level:" followed by an empty white input field. To its right, a red error message reads "Data Required. Please Complete."
- "GradeFee:" followed by an input field containing "10\$". To its right, a red error message reads "Please Remove Symbol Character."

2.17[vi][A][2] Screenshot E {Punctuation Present}

The screenshot shows a Windows application window titled "frmAddField". The title bar features a logo of a guitar and the text "The Mitchell School Of Music". The main title of the window is "Add Grade". A blue header bar contains two buttons: "Add New Grade" and "Return to Grade Table". The main area has two input fields: "Grade Level:" and "GradeFee:". The "Grade Level:" field is empty and has a red validation message: "Data Required. Please Complete.". The "GradeFee:" field contains the value "10@" and has a red validation message: "Please Remove Punctuation."

2.17[vi][A][2] Screenshot F {Range}

The screenshot shows the same Windows application window as in Screenshot E. The "Grade Level:" field is still empty with the validation message "Data Required. Please Complete.". The "GradeFee:" field now contains the value "100" and has a red validation message: "Value Exceeds limit. Please stay within the range of (1 - 20)."

*Process [B]) Update Display***2.17[vi][B][1] – Grade Record Display****2.17[vi][B][1] Screenshot A****2.17[vi][B][1] Screenshot B**

2.17[viii] Scheduled Lesson Record**Process [A]) Validation****2.17[viii][A][1] – Student ID Validation****2.17[viii][A][1] Screenshot A**

The screenshot shows a Windows application window titled "Schedule New Lesson". The title bar includes the logo of "The Michael School Of Music" and the window title "frmAddField". Below the title bar is a toolbar with icons for "Schedule New Lesson" and "Return to Scheduled Lessons Table". The main area contains several input fields and a dropdown menu:

- Student ID:** A dropdown menu is open, showing a list of student names and IDs. The entry "16 | Lucy Dysart" is highlighted.
- Teacher ID:** A dropdown menu is open, showing a list of teacher names and IDs. The entry "1 | Sandra Smith" is visible.
- Purchased Lesson ID:** An empty dropdown menu.
- No. of weeks:** A dropdown menu with the option "5 Weeks" selected.
- Start Date:** An empty text input field.
- Booked Time:** A dropdown menu with the option "13:00" selected.
- Booked Day(s):** An empty dropdown menu.

2.17[viii][A][1] Screenshot B

The screenshot shows the same "Schedule New Lesson" window as in Screenshot A, but with different data entered into the fields:

- Student ID:** The dropdown menu is closed, showing the selected value "16 | Lucy Dysart".
- Teacher ID:** The dropdown menu is closed, showing the selected value "1 | Sandra Smith".
- Purchased Lesson ID:** An empty dropdown menu.
- No. of weeks:** A dropdown menu with the option "5 Weeks" selected.
- Start Date:** An empty text input field.
- Booked Time:** A dropdown menu with the option "13:00" selected.
- Booked Day(s):** An empty dropdown menu.

2.17[viii][A][2] – Teacher ID Validation**2.17[viii][A][2] Screenshot A**

The screenshot shows a Windows application window titled "frmAddField". The title bar features a logo of a guitar and the text "The Musical School Of Music". The main title of the form is "Schedule New Lesson".

The form contains the following fields and controls:

- Student ID:** A dropdown menu showing "16 | Lucy Dysart".
- Teacher ID:** A dropdown menu showing "1 | Sandra Smith".
- Purchased Lesson ID:** A dropdown menu showing "1 | Sandra Smith", "2 | Carey Gamble", "3 | Donna Shaw", "4 | Nathan Jones", "5 | Gary Gibson", and "6 | Jamie Dome".
- No. of weeks:** A dropdown menu showing "No. of weeks" and "5 Weeks".
- Start Date:** A date picker control.
- Booked Time:** A dropdown menu showing "13:00".
- Booked Day(s):** A day picker control.

At the top right of the form is a red-bordered button labeled "Return to Scheduled Lessons Table".

2.17[viii][A][2] Screenshot B

The screenshot shows the same Windows application window as Screenshot A, titled "frmAddField". The title bar features the same logo and text.

The main title of the form is "Schedule New Lesson".

The form contains the following fields and controls:

- Student ID:** A dropdown menu showing "16 | Lucy Dysart".
- Teacher ID:** A dropdown menu showing "4 | Nathan Jones".
- Purchased Lesson ID:** An empty dropdown menu.
- No. of weeks:** A dropdown menu showing "No. of weeks" and "5 Weeks".
- Start Date:** A date picker control.
- Booked Time:** A dropdown menu showing "13:00".
- Booked Day(s):** A day picker control.

At the top right of the form is a red-bordered button labeled "Return to Scheduled Lessons Table".

2.17[viii][A][2] – Purchase Lesson D Validation**2.17[viii][A][3] Screenshot A**

frmAddField

The Mitchell School Of Music

Schedule New Lesson

[Schedule New Lesson](#) [Return to Scheduled Lessons Table](#)

LessonsPurchased	StudentID	LessonBundleID	Purchased_Date
8	16	2	01/08/2016
*			

Student ID: 16 | Lucy Dysart

Teacher ID: 4 | Nathan Jones

Purchased Lesson ID:

No. of weeks: 5 Weeks

Start Date:

Booked Time: 13:00

Booked Day(s):

2.17[viii][A][3] Screenshot B

frmAddField

The Mitchell School Of Music

Schedule New Lesson

[Schedule New Lesson](#) [Return to Scheduled Lessons Table](#)

Student ID: 16 | Lucy Dysart

Teacher ID: 4 | Nathan Jones

Purchased Lesson ID:

No. of weeks: 5 Weeks

Start Date:

Booked Time: 13:00

Booked Day(s):

2.17[viii][A][2] – Number Of Week Validation**2.17[viii][A][4] Screenshot A**

The screenshot shows a Windows application window titled "frmAddField". The main title bar says "Schedule New Lesson". In the top left corner, there is a logo of a guitar and the text "The Mitchell School Of Music". The interface includes two buttons: "Schedule New Lesson" on the left and "Return to Scheduled Lessons Table" on the right. The form contains several input fields and dropdown menus:

- Student ID:** 16 | Lucy Dysart (dropdown menu)
- Teacher ID:** 4 | Nathan Jones (dropdown menu)
- Purchased Lesson ID:** (text input field with a calendar icon)
- No. of weeks:** 5 Weeks (dropdown menu)
 - 5 Weeks
 - 10 Weeks
 - 15 Weeks
 - 20 Weeks** (selected option)
 - 30 Weeks
- Start Date:** (text input field with a calendar icon)
- Booked Time:** (dropdown menu)
 - 20 Weeks
 - 30 Weeks
- Booked Day(s):** (text input field with a days-of-the-week icon)

2.17[viii][A][4] Screenshot B

This screenshot is identical to Screenshot A, showing the "Schedule New Lesson" form. The only difference is the value selected in the "No. of weeks:" dropdown menu, which has been changed from "20 Weeks" to "30 Weeks". All other fields and their values remain the same as in Screenshot A.

2.17[viii][A][2] – Start Date Validation**2.17[viii][A][5] Screenshot A**

frmAddField

The Musical School Of Music

Schedule New Lesson

Schedule New Lesson Return to Scheduled Lessons Table

Student ID: 16 | Lucy Dysart

Teacher ID: 4 | Nathan Jones

Purchased Lesson ID:

No. of weeks: 20 Weeks

Start Date:

Booked Time: 13:00

Booked Day(s):

April 2017

Mon	Tue	Wed	Thu	Fri	Sat	Sun
27	28	29	30	31	1	2
3	4	5	6	7	8	9
11	12	13	14	15	16	
17	18	19	20	21	22	23
24	25	26	27	28	29	30
1	2	3	4	5	6	7

Today: 25/04/2017

2.17[viii][A][5] Screenshot B

frmAddField

The Musical School Of Music

Schedule New Lesson

Schedule New Lesson Return to Scheduled Lessons Table

Student ID: 16 | Lucy Dysart

Teacher ID: 4 | Nathan Jones

Purchased Lesson ID:

No. of weeks: 20 Weeks

Start Date: 20/04/2017 

Booked Time: 13:00

Booked Day(s):

2.17[viii][A][2] – Booked Days Validation**2.17[viii][A][6] Screenshot A**

The screenshot shows the 'Schedule New Lesson' window with the following fields filled:

- Student ID:** 16 | Lucy Dysart
- Teacher ID:** 4 | Nathan Jones
- Purchased Lesson ID:** [empty]
- No. of weeks:** 20 Weeks
- Start Date:** 20/04/2017
- Booked Time:** 13:00
- Booked Day(s):** [dropdown menu open]
 - Monday
 - Tuesday
 - Wednesday
 - Thursday
 - Friday

2.17[viii][A][2] – Booked Time Validation**2.17[viii][A][7] Screenshot A**

The screenshot shows the 'Schedule New Lesson' window with the following fields filled:

- Student ID:** 4 | Linda This is a test Si
- Teacher ID:** 1 | Sandra Smith
- Purchased Lesson:** [empty]
- No. of weeks:** 5 Weeks
- Start Date:** [empty]
- Booked Time:** [dropdown menu open]
 - 13:00
 - 13:30
 - 14:00
 - 14:30
 - 15:00
 - 15:30
 - 16:00
 - 16:30
 - 17:00** [selected]
 - 17:30
 - 18:00
 - 18:30
 - 19:00
 - 19:30
 - 20:00
 - 20:30
- Booked Day(s):** [dropdown menu open]
 - 13:00
 - 13:30
 - 14:00
 - 14:30
 - 15:00
 - 15:30
 - 16:00
 - 16:30
 - 17:00** [selected]
 - 17:30
 - 18:00
 - 18:30
 - 19:00
 - 19:30
 - 20:00
 - 20:30

2.17[viii][A][7] Screenshot B

Schedule New Lesson

Schedule New Lesson [Return to Scheduled Lessons Table](#)

Student ID:	4 Linda This is a test Si
Teacher ID:	1 Sandra Smith
Purchased Lesson:	<input type="button" value=""/>
No. of weeks:	5 Weeks
Start Date:	<input type="button" value=""/>
Booked Time:	17:00
Booked Day(s):	<input type="button" value=""/>

2.17[viii][A][2] – End Date Validation**2.17[viii][A][8] Screenshot A**

Schedule New Lesson

Schedule New Lesson [Return to Scheduled Lessons Table](#)

Schedule ID:	11
Student ID:	4 Linda This is a test Si
Teacher ID:	1 Sandra Smith
Purchased Lesson:	<input type="button" value=""/>
No. of weeks:	5 Weeks
Start Date:	28/10/2016 00:00:00
Booked Time:	13:00
Booked Day(s):	Monday
End Date:	26/05/2017 00:00:00

April 2017

Mon	Tue	Wed	Thu	Fri	Sat	Sun
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
1	2	3	4	5	6	7

Today: 25/04/2017

2.17[viii][A][8] Screenshot B

Schedule New Lesson

Schedule ID: 11

Student ID: 4 | Linda This is a test Si

Teacher ID: 1 | Sandra Smith

Purchased Lesson:

No. of weeks: 5 Weeks

Start Date: 28/10/2016 00:00:00

Booked Time: 13:00

Booked Day(s): Monday

End Date: 12/04/2017

[Schedule New Lesson](#) [Return to Scheduled Lessons Table](#)

Process [B]) Update Display**2.17[vii][B][1] – Scheduled Lesson Record Validation****2.17[viii][B][1] Screenshot A**

Scheduled Lessons List

StudentID: 6

Student Details

First Name: Daniel Waters

Contact Number: 37719048

Teacher Details

TeacherID: 1

First Name: Sandra Smith

Specialisation: String

Room: 1

Grade Details

GradeID: 4

Grade: Diploma

ScheduleID: 11

ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date
11	6	1	3	30 Weeks	28/10/2016	Monday	09:30	26/05/2017

Search for Student: Adam Ackles

Add new Student

Update Lesson Record

Remove Lesson Record

Return to Calendar [Placeholder]

Return to Main Menu

2.17[viii][B][1] Screenshot B
2.17[ix] Additional Functions**Process [A]) Functions****2.17[ix][A][1] – Add information to End Dates****2.17[ix][A][1] Screenshot A**

frmClassSchedule.cs [Design]

dbo.Scheduled_Lessons [Data] dbo.LessonsPurchased [Data] dbo.LessonDates [Data] dbo.LessonDate_Archive [Data] dbo.Scheduled_Lessons [Data]

	ScheduleID	StudentID	TeacherID	LessonsPurcha...	Number_Of_W...	Start_Date	Booked_Day(s)	Booked_Time	End_Date
6	7	2	1	10 Weeks	26/10/2018	Monday ... 09:30			20/01/1990
7	7	5	2	10 Weeks	24/10/2017	Tuesday + Wed... 09:30			02/01/2018
8	10	4	33	30 Weeks	27/10/2017	Thursday ... 09:30			25/05/2018
11	6	1	13	30 Weeks	28/10/2017	Monday ... 09:30			26/05/2018
12	16	1	8	5 Weeks	25/10/2017	Monday + Frid... 11:30			29/11/2017
13	12	3	1	10 Weeks	24/10/2017	Tuesday ... 12:00			02/01/2018
14	22	2	2	15 Weeks	28/10/2017	Thursday + Frid... 13:00			10/02/2018
15	5	3	1	5 Weeks	25/10/2017	Monday + Tues... 14:30			29/11/2017
16	17	4	2	15 Weeks	27/10/2017	Wednesday + T... 16:00			09/02/2018
17	4	2	1	10 Weeks	05/01/2017	Friday ... 10:30			16/03/2017
18	8	3	2	20 Weeks	28/10/2017	Thursday ... 11:00			17/03/2018
19	5	2	2	20 Weeks	24/10/2017	Friday ... 13:00			13/03/2018
20	21	3	1	5 Weeks	26/10/2017	Monday + Tues... 12:00			30/11/2017
21	19	3	1	10 Weeks	28/10/2017	Wednesday ... 13:00			06/01/2018
22	6	1	1	10 Weeks	26/10/2017	Thursday ... 10:00			04/01/2018
25	13	2	3	30 Weeks	01/11/2016	Monday ... 15:00			30/05/2017
	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

2.17[i][A][i] Screenshot B

Schedule New Lesson

Return to Scheduled Lessons Table

	LessonsPurchased	S
▶	31	4
*		
<		
>		

2.17[i][A][i] Screenshot C

Search Solution Explorer (Ctrl+F)

Object Explorer

LessonDateID	ScheduleID	Date	Attended
1	8	27/10/2016 00:00:00	False
2	8	03/11/2016 00:00:00	False
3	8	10/11/2016 00:00:00	False
4	8	17/11/2016 00:00:00	False
5	8	24/11/2016 00:00:00	False
6	8	01/12/2016 00:00:00	False
7	8	08/12/2016 00:00:00	False
8	8	15/12/2016 00:00:00	False
9	8	22/12/2016 00:00:00	False
10	8	29/12/2016 00:00:00	False
11	8	05/01/2017 00:00:00	False
12	8	12/01/2017 00:00:00	False
13	8	19/01/2017 00:00:00	False
14	8	26/01/2017 00:00:00	False
15	8	02/02/2017 00:00:00	False
16	8	09/02/2017 00:00:00	False
17	8	16/02/2017 00:00:00	False
18	8	23/02/2017 00:00:00	False
19	8	30/03/2017 00:00:00	False
20	8	06/04/2017 00:00:00	False
21	8	13/04/2017 00:00:00	False
22	8	20/04/2017 00:00:00	False
23	8	27/04/2017 00:00:00	False
24	8	04/05/2017 00:00:00	False
25	8	11/05/2017 00:00:00	False
26	8	18/05/2017 00:00:00	False
27	8	31/10/2016 00:00:00	False
28	11	07/11/2016 00:00:00	False
29	11	14/11/2016 00:00:00	False
30	11	21/11/2016 00:00:00	False
31	11	28/11/2016 00:00:00	False
32	11	05/12/2016 00:00:00	False
33	11	12/12/2016 00:00:00	False
34	11	19/12/2016 00:00:00	False
35	11	26/12/2016 00:00:00	False
36	11	02/01/2017 00:00:00	False
37	11	09/01/2017 00:00:00	False
38	11	16/01/2017 00:00:00	False

2.17[ix][A][2] – Automatically calculate Bundle Costs**2.17[ix][A][2] Screenshot A**

LessonsPurchasedID	StudentID	LessonBundleID	Purchased_Date	Payment_Method	Payment_Received	Total_Bundle_Cost	Scheduled
1	5	1	01/08/2016	Cash	True	01/08/2016	0
2	5	2	08/08/2016	Pending	... False	.../.../...	True
3	9	3	02/08/2016	Cash	True	02/08/2016	0
4	7	15	03/08/2016	Card	True	03/08/2016	0
5	16	3	01/08/2016	Cash	True	03/08/2016	0
6	8	1	01/08/2016	Card	True	03/08/2016	0
7	14	1	16/08/2016	Card	True	05/08/2016	0
8	9	2	01/08/2016	Pending	... False	.../.../...	False
9	6	3	12/08/2016	Cash	True	12/08/2016	0
10	15	3	10/08/2016	Cash	True	12/08/2016	0
11	18	2	01/08/2016	Card	True	02/08/2016	0
12	20	1	01/08/2016	Card	True	08/08/2016	0
13	22	3	04/08/2016	Cash	True	04/08/2016	0
14	17	2	01/08/2016	Pending	... False	.../.../...	False
15	21	2	09/08/2016	Cash	True	09/08/2016	0
16	25	1	11/08/2016	Cash	True	16/08/2016	0
17	13	2	10/08/2016	Cash	True	11/08/2016	0
18	11	1	01/08/2016	Pending	... False	.../.../...	False
19	10	1	01/09/2016	Card	True	02/09/2016	0
20	26	2	05/08/2016	Card	True	05/08/2016	0
21	24	3	05/08/2016	Cash	True	05/08/2016	0
22	22	1	01/08/2016	Cash	True	01/08/2016	0
23	17	1	08/08/2016	Card	True	09/08/2016	0
24	19	2	11/08/2016	Pending	... False	.../.../...	False
25	8	1	14/08/2016	Card	True	14/09/2016	0
26	4	3	01/08/2016	Cash	True	01/09/2016	0
27	5	2	10/08/2016	Pending	... False	.../.../...	False
28	10	3	01/08/2016	Pending	... False	.../.../...	False
29	7	1	01/08/2016	Card	True	07/10/2016	0
30	9	2	13/08/2016	Cash	True	13/09/2016	0
31	14	3	16/08/2016	Pending	... False	.../.../...	False
32	18	1	08/09/2016	Card	True	08/09/2016	0
33	19	2	12/09/2016	Pending	... False	.../.../...	False
34	NULL	NULL	NULL	NULL	NULL	NULL	NULL

2.17[ix][A][2] Screenshot B

The Mitchell School Of Music

New Lesson Purchased:

Purchase New Lesson **Return to Purchased Lessons Table**

Student ID:

Purchase ID:

Purchase Date:

Payment Method:

Payment Received:

payment Receipt Date:

2.17[ix][A][2] Screenshot C

dbo.Scheduled_Lessons [Data]		dbo.LessonsPurchased [Data]		dbo.LessonDates [Data]		dbo.LessonDate_Archive [Data]		dbo.Scheduled_Lessons [Data]		frmCalenderDates.cs #
ScheduledLessonID	LessonsPurchasedID	StudentID	LessonBundleID	Purchased_Date	Payment_Method	Payment_Received	Payment_Received	Total_Bundle_Cost	Scheduled	
1	5	1	01/08/2016	Cash	True	01/08/2016	100	False		
5	5	2	08/08/2016	Pending	... False	.../.../....	190	True		
6	9	3	02/08/2016	Cash	True	02/08/2016	675	False		
7	15	3	03/08/2016	Card	True	03/08/2016	540	False		
8	16	3	01/08/2016	Cash	True	03/08/2016	675	False		
9	8	1	07/08/2016	Card	True	03/08/2016	250	False		
10	14	1	16/08/2016	Card	True	05/08/2016	150	False		
11	9	2	01/08/2016	Pending	... False	.../.../....	475	False		
13	6	3	12/08/2016	Cash	True	12/08/2016	675	False		
14	15	3	10/08/2016	Cash	True	12/08/2016	540	False		
15	18	2	07/08/2016	Card	True	02/08/2016	475	False		
17	20	1	01/08/2016	Card	True	08/08/2016	100	False		
18	22	3	04/08/2016	Cash	True	04/08/2016	405	False		
19	17	2	01/08/2016	Pending	... False	.../.../....	380	False		
20	21	2	09/08/2016	Cash	True	09/08/2016	285	False		
21	25	1	11/08/2016	Cash	True	16/08/2016	250	False		
22	13	2	10/08/2016	Cash	True	11/08/2016	380	False		
23	11	1	01/08/2016	Pending	... False	.../.../....	100	False		
24	10	1	01/09/2016	Card	True	02/09/2016	100	False		
25	26	2	05/08/2016	Card	True	05/08/2016	190	False		
26	24	3	05/08/2016	Cash	True	05/08/2016	270	False		
27	22	1	01/08/2016	Cash	True	01/08/2016	150	False		
28	17	1	08/08/2016	Card	True	09/08/2016	200	False		
29	19	2	11/08/2016	Pending	... False	.../.../....	380	False		
30	8	1	14/08/2016	Card	True	14/09/2016	250	False		
31	4	3	01/08/2016	Cash	True	01/09/2016	405	False		
32	5	2	10/08/2016	Pending	... False	.../.../....	190	False		
33	10	3	01/08/2016	Pending	... False	.../.../....	270	False		
34	7	1	01/08/2016	Card	True	07/10/2016	150	False		
35	9	2	13/08/2016	Cash	True	13/09/2016	475	False		
36	14	3	16/08/2016	Pending	... False	.../.../....	405	False		
37	18	1	08/09/2016	Card	True	08/09/2016	250	False		
38	19	2	12/09/2016	Pending	... False	.../.../....	380	False		
o	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	

2.17[ix][A][3] – Automatically calculate End Dates**2.17[ix][A][3] Screenshot A**

dbo.Scheduled_Lessons [Data]		dbo.LessonsPurchased [Data]		dbo.LessonDates [Data]		dbo.LessonDate_Archive [Data]		dbo.Scheduled_Lessons [Data]		
ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date		
6	7	2	1	10 Weeks	26/10/2018	Monday	... 09:30	20/01/1990		
7	7	5	2	10 Weeks	24/10/2017	Tuesday + Wed...	09:30	02/01/2018		
8	10	4	33	30 Weeks	27/10/2017	Thursday	... 09:30	25/05/2018		
11	6	1	13	30 Weeks	28/10/2017	Monday	... 09:30	26/05/2018		
12	16	1	8	5 Weeks	25/10/2017	Monday + Frid...	11:30	29/11/2017		
13	12	3	1	10 Weeks	24/10/2017	Tuesday	... 12:00	02/01/2018		
14	22	2	2	15 Weeks	28/10/2017	Thursday + Frid...	13:00	10/02/2018		
15	5	3	1	5 Weeks	25/10/2017	Monday + Tues...	14:30	29/11/2017		
16	17	4	2	15 Weeks	27/10/2017	Wednesday + T...	16:00	09/02/2018		
17	4	2	1	10 Weeks	05/01/2017	Friday	... 10:30	16/03/2017		
18	8	3	2	20 Weeks	28/10/2017	Thursday	... 11:00	17/03/2018		
19	5	2	2	20 Weeks	24/10/2017	Friday	... 13:00	13/03/2018		
20	21	3	1	5 Weeks	26/10/2017	Monday + Tues...	12:00	30/11/2017		
21	19	3	1	10 Weeks	28/10/2017	Wednesday	... 13:00	06/01/2018		
22	6	1	1	10 Weeks	26/10/2017	Thursday	... 10:00	04/01/2018		
25	13	2	3	30 Weeks	01/11/2016	Monday	... 15:00	30/05/2017		
o	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	

2.17[i][A][3] Screenshot B

Schedule New Lesson

Return to Scheduled Lessons Table

	LessonsPurchased
>	31
*	4
<	

2.17[i][A][3] Screenshot C

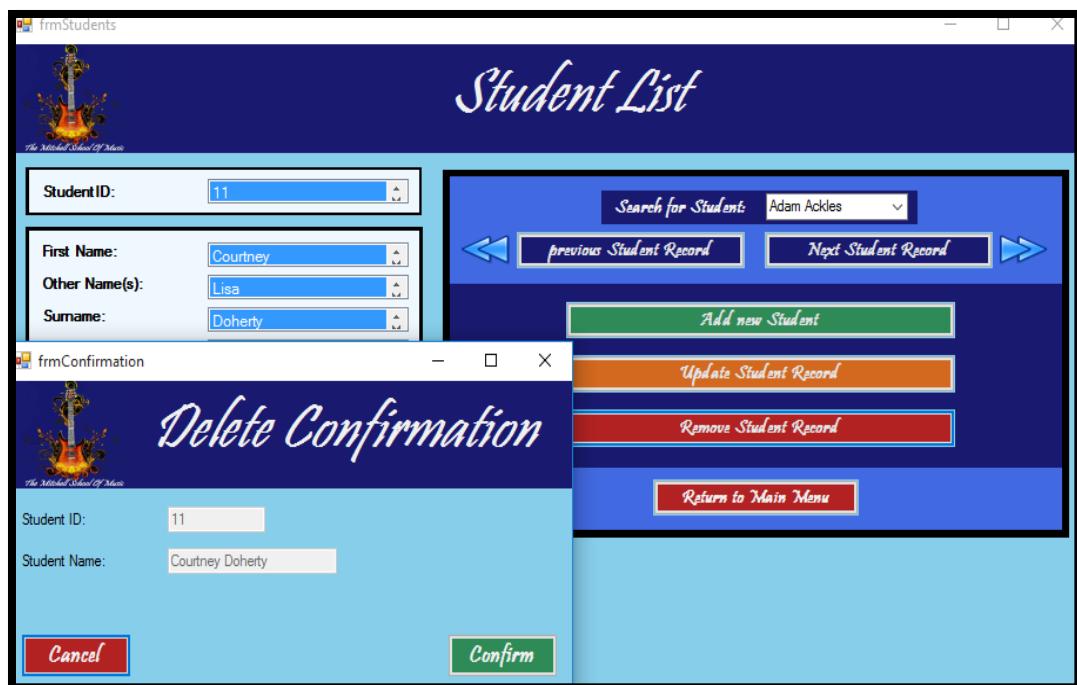
dbo.Scheduled_Lessons [Data] X dbo.LessonsPurchased [Data] dbo.LessonDates [Data] dbo.LessonDate_Archive [Data] dbo.Scheduled_Lessons [Data]

Max Rows: 1000

	ScheduleID	StudentID	TeacherID	LessonsPurcha...	Number_Of_W...	Start_Date	Booked_Day(s)	Booked_Time	End_Date
D	7	7	5	2	10 Weeks	24/10/2017	Tuesday + Wed...	09:30	02/01/2018
	8	10	4	33	30 Weeks	27/10/2017	Thursday ...	09:30	25/05/2018
	11	6	1	13	30 Weeks	28/10/2017	Monday ...	09:30	26/05/2018
	12	16	1	8	5 Weeks	25/10/2017	Monday + Frid...	11:30	29/11/2017
	13	12	3	1	10 Weeks	24/10/2017	Tuesday ...	12:00	02/01/2018
	14	22	2	2	15 Weeks	28/10/2017	Thursday + Frid...	13:00	10/02/2018
	15	5	3	1	5 Weeks	25/10/2017	Monday + Tues...	14:30	29/11/2017
	16	17	4	2	15 Weeks	27/10/2017	Wednesday + T...	16:00	09/02/2018
	18	8	3	2	20 Weeks	28/10/2017	Thursday ...	11:00	17/03/2018
	19	5	2	2	20 Weeks	24/10/2017	Friday ...	13:00	13/03/2018
	20	21	3	1	5 Weeks	26/10/2017	Monday + Tues...	12:00	30/11/2017
	21	19	3	1	10 Weeks	28/10/2017	Wednesday ...	13:00	06/01/2018
	22	6	1	1	10 Weeks	26/10/2017	Thursday ...	10:00	04/01/2018
	25	13	2	3	30 Weeks	01/11/2016	Monday ...	15:00	30/05/2017
O	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

2.18) Confirmation Form

Process [A])

2.18[A][1] – Selected Record Display**2.18[A][1] Screenshot A****2.18[A][1] Screenshot B**

2.18[A][2] – Confirm Button**2.18[A][2] Screenshot A**

Add Student

Add New Student **Return to Students Table**

First Name: TEST
 Other Name(s): TEST
 Surname: TEST
 Date Of Birth: 14/04/2017
 Address: 123 REST
 Town: TEST
 PostCode: BT546WE
 Contact Number: 02753454353
 Email Address: TEST@TEST.COM
 GradeID: 1
 InstrumentID: 1
 Tuition Fee Received

2.18[A][2] Screenshot B

Student List

StudentID: 32
 First Name: TEST
 Other Name(s): TEST
 Surname: TEST

Search for Student: Adam Ackles

previous Student Record Next Student Record

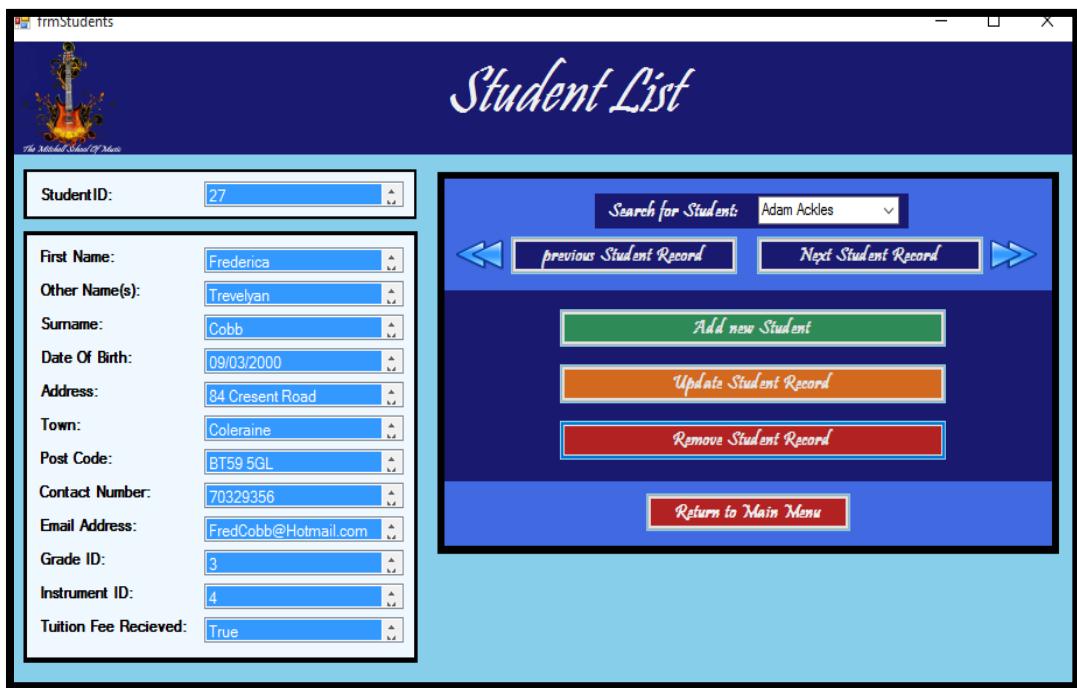
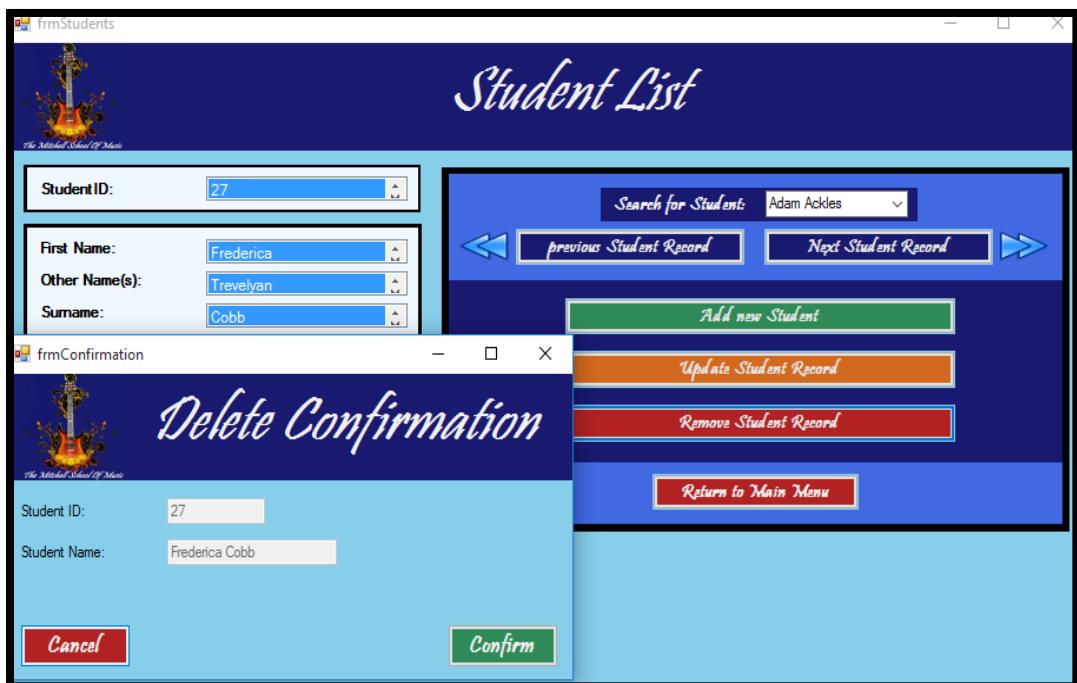
Add new Student Update Student Record Remove Student Record Return to Main Menu

Delete Confirmation

Student ID: 32
 Student Name: TEST TEST

frmConfirmation

Cancel **Confirm**

2.18[A][2] Screenshot C**2.18[A][3] – Return Button****2.18[A][3] Screenshot A**

2.18[A][3] Screenshot B

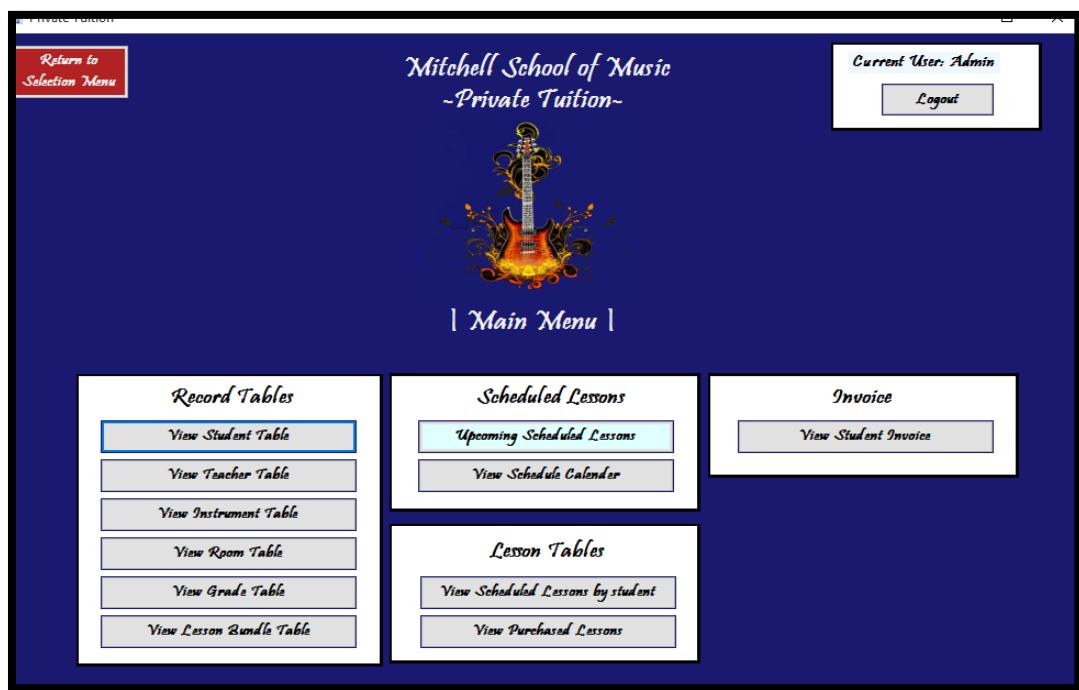
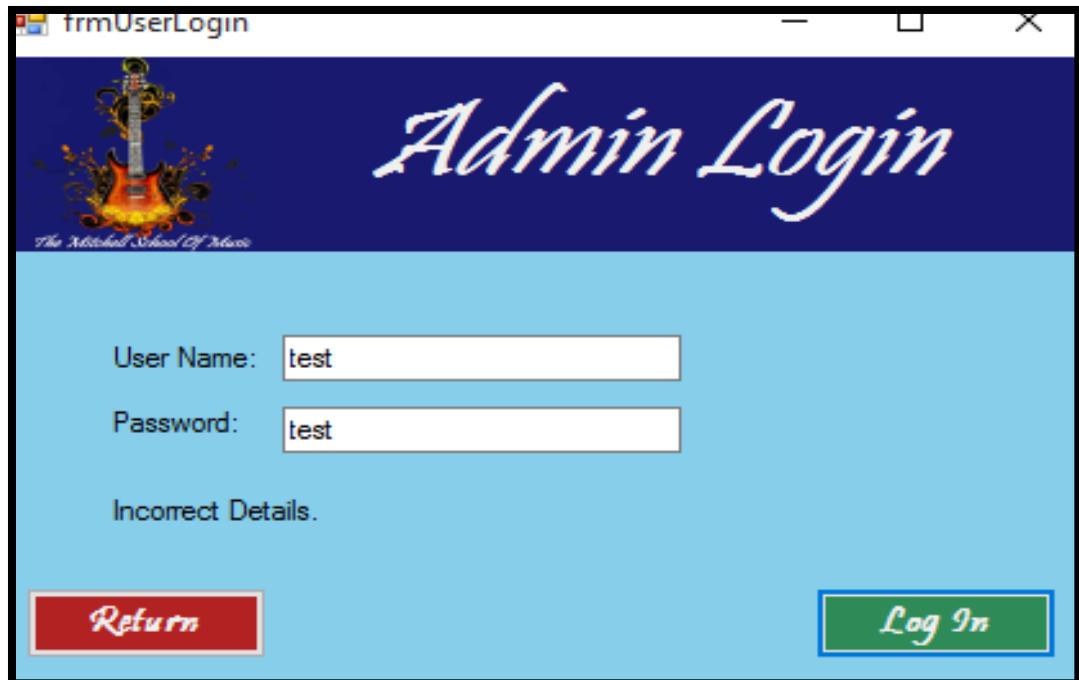
The screenshot shows a Windows application window titled "frmStudents". The title bar features a logo of a stylized guitar and the text "The National School Of Music". The main title "Student List" is displayed prominently at the top center. On the left side, there is a vertical stack of text boxes for entering student information, each with a label and a dropdown menu. The fields include: Student ID (27), First Name (Frederica), Other Name(s) (Trevelyan), Surname (Cobb), Date Of Birth (09/03/2000), Address (84 Crescent Road), Town (Coleraine), Post Code (BT59 5GL), Contact Number (70329356), Email Address (FredCobb@Hotmail.com), Grade ID (3), Instrument ID (4), and Tuition Fee Received (True). On the right side, there is a search bar labeled "Search for Student" with a dropdown menu set to "Adam Ackles". Below the search bar are four buttons: "previous Student Record" (left arrow), "Next Student Record" (right arrow), "Add new Student" (green button), "Update Student Record" (orange button), "Remove Student Record" (red button), and "Return to Main Menu" (blue button).

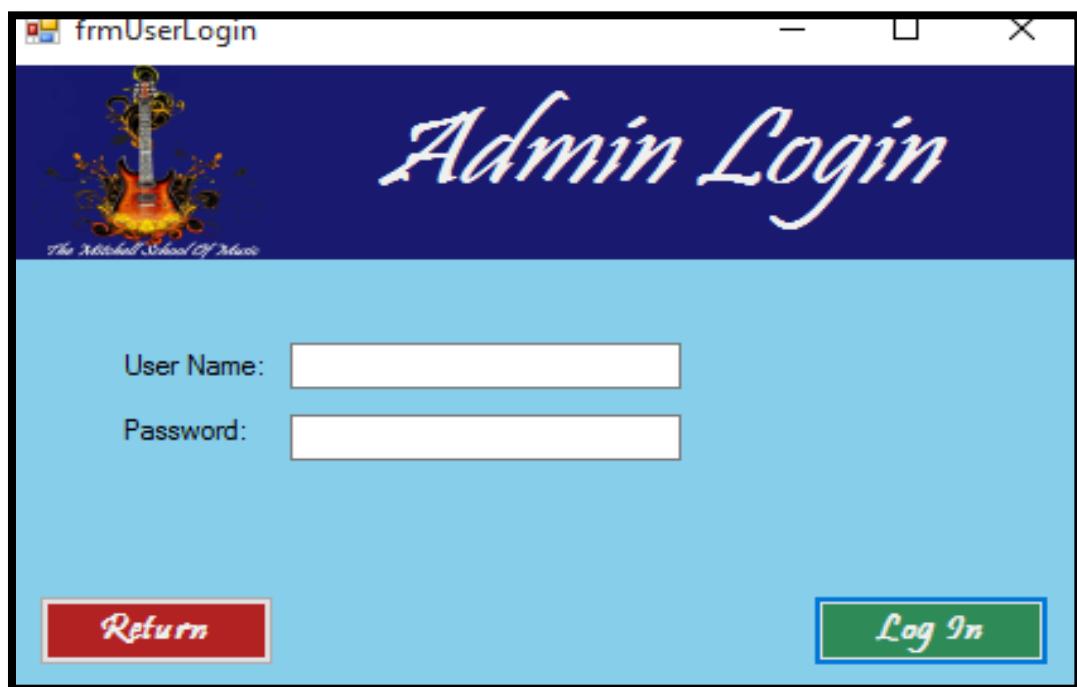
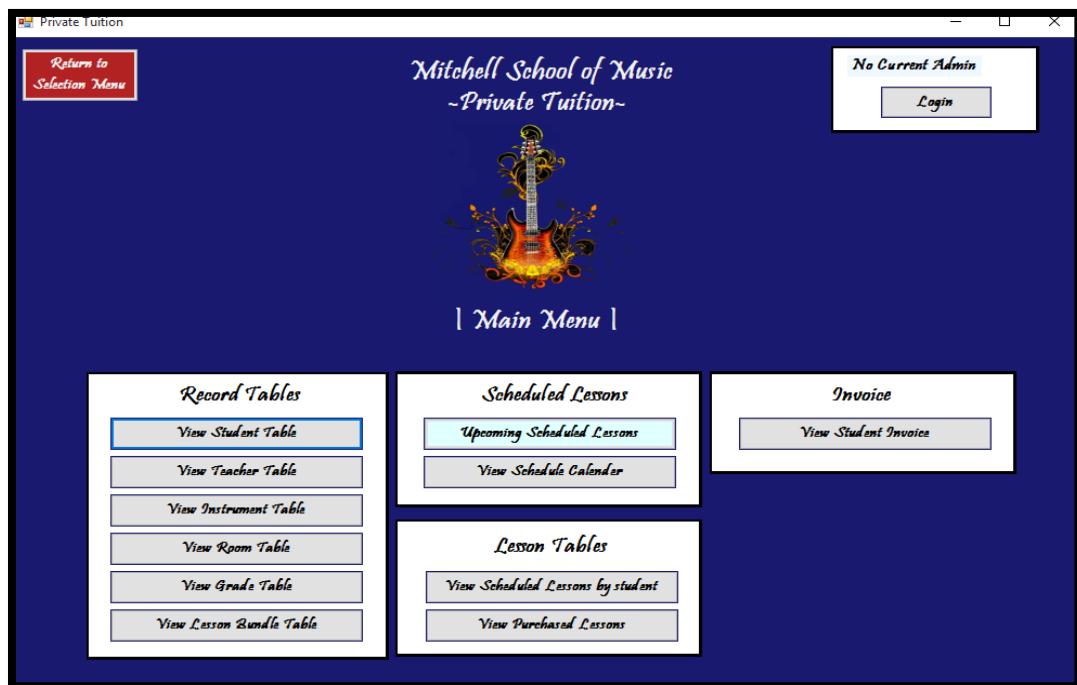
2.19) User Login

Process [A]) Form Display

2.19[A][1] – User Login (Correct Details)**2.19[A][1] Screenshot A**

The screenshot shows a Windows application window titled "frmUserLogin". The title bar features a logo of a stylized guitar and the text "The National School Of Music". The main title "Admin Login" is displayed prominently at the top center. The form contains two text input fields: "User Name" (Admin) and "Password" (Admin). Below the password field, the message "Incorrect Details." is displayed in red. At the bottom of the screen, there are two buttons: "Return" (red button) on the left and "Log In" (green button) on the right.

2.19[A][1] Screenshot B**2.19[A][2] – User Login (Incorrect Details)****2.19[A][2] Screenshot A**

2.19[A][3] – Return Button**2.19[A][3] Screenshot A****2.19[A][3] Screenshot B**

2.20) Invoice**Process [A]) Form Display****2.20[A][1] – Purchased Lesson Record****2.20[A][1] Screenshot A**

The screenshot shows a software window titled "Student Invoice". At the top, it says "Selected Student: Linda Simmons". Below this is a table with four columns: "Purchased Lesson ID:", "Lesson Bundle:", "Purchased Date:", and "Bundle Cost:". There are two rows of data:

Purchased Lesson ID:	Lesson Bundle:	Purchased Date:	Bundle Cost:
23	1	01/08/2016	100
33	3	01/08/2016	270

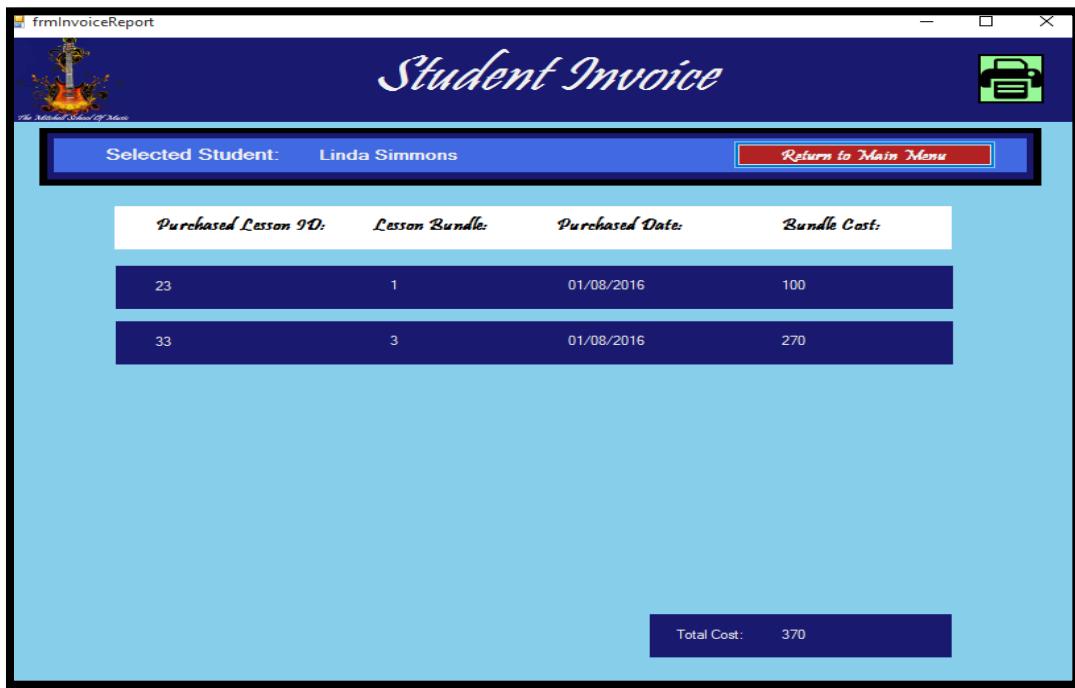
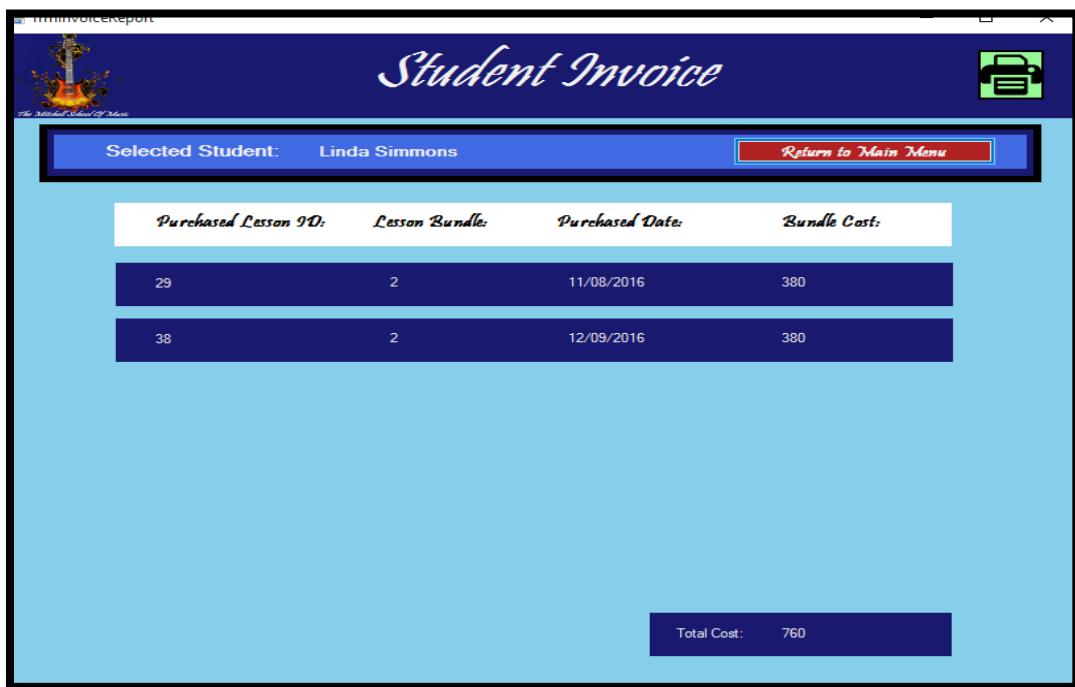
At the bottom right, it says "Total Cost: 370".

2.20[A][1] Screenshot B

The screenshot shows a software window titled "Student Invoice". At the top, it says "Selected Student: Linda Simmons". Below this is a table with four columns: "Purchased Lesson ID:", "Lesson Bundle:", "Purchased Date:", and "Bundle Cost:". There are two rows of data:

Purchased Lesson ID:	Lesson Bundle:	Purchased Date:	Bundle Cost:
29	2	11/08/2016	380
38	2	12/09/2016	380

At the bottom right, it says "Total Cost: 760".

2.20[A][2] – Total Bundle Cost**2.20[A][2] Screenshot A****2.20[A][2] Screenshot B**

2.20[A][3] – Selected Student Name**2.20[A][3] Screenshot A**

The screenshot shows a software window titled "Student Invoice" from "The Method School Of Music". The selected student is Linda Simmons. The purchase history table lists two entries:

Purchased Lesson ID:	Lesson Bundle:	Purchased Date:	Bundle Cost:
23	1	01/08/2016	100
33	3	01/08/2016	270

Total Cost: 370

2.20[A][3] Screenshot B

The screenshot shows a software window titled "Student Invoice" from "The Method School Of Music". The selected student is Linda Simmons. The purchase history table lists two entries:

Purchased Lesson ID:	Lesson Bundle:	Purchased Date:	Bundle Cost:
29	2	11/08/2016	380
38	2	12/09/2016	380

Total Cost: 760

2.20[A][4] – Error Message**2.20[A][4] Screenshot A**

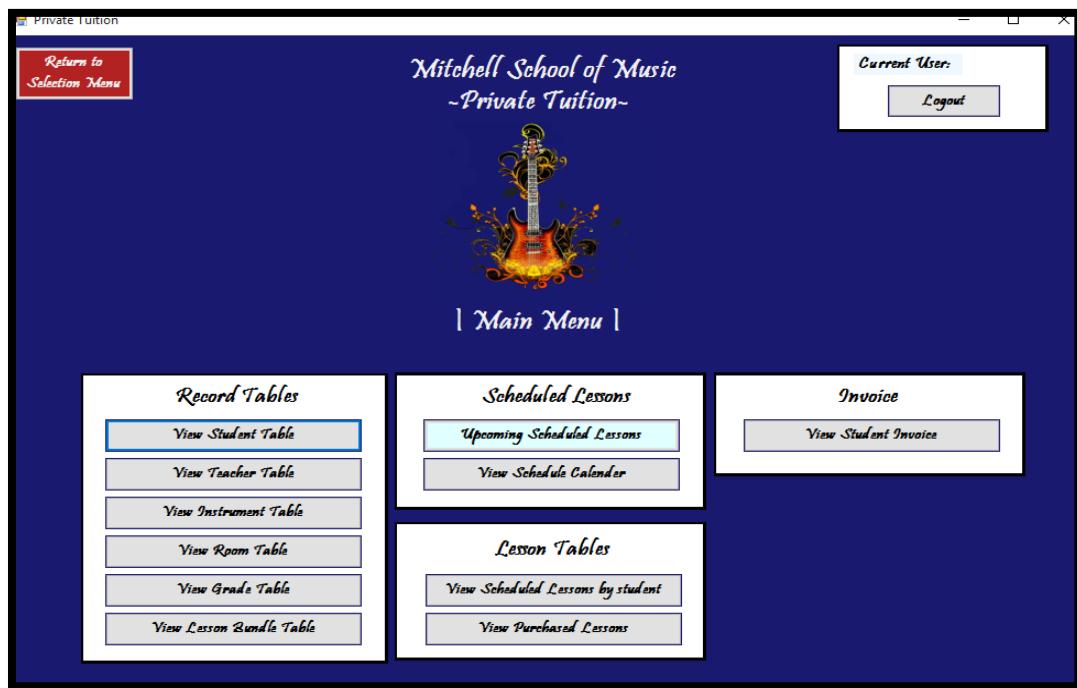
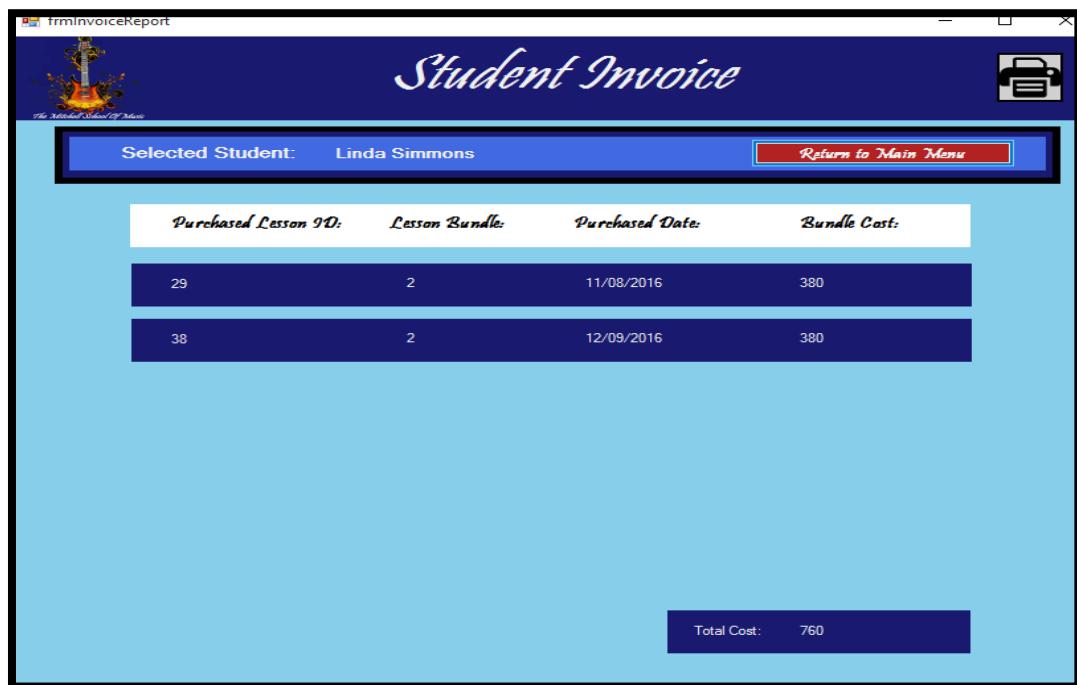
The screenshot shows a Windows application window titled "frmInvoiceReport". The main title bar says "Student Invoice". In the top left corner, there is a logo of a guitar and the text "The Method School Of Music". On the right side of the title bar is a printer icon. Below the title bar, a blue header bar displays "Selected Student: Linda Simmons" and a red "Return to Main Menu" button. The main body of the window is light blue and contains the text: "This Student currently possess no outstanding purchased lesson records." At the bottom right, there is a dark blue footer bar with the text "Total Cost: [Place Holder]".

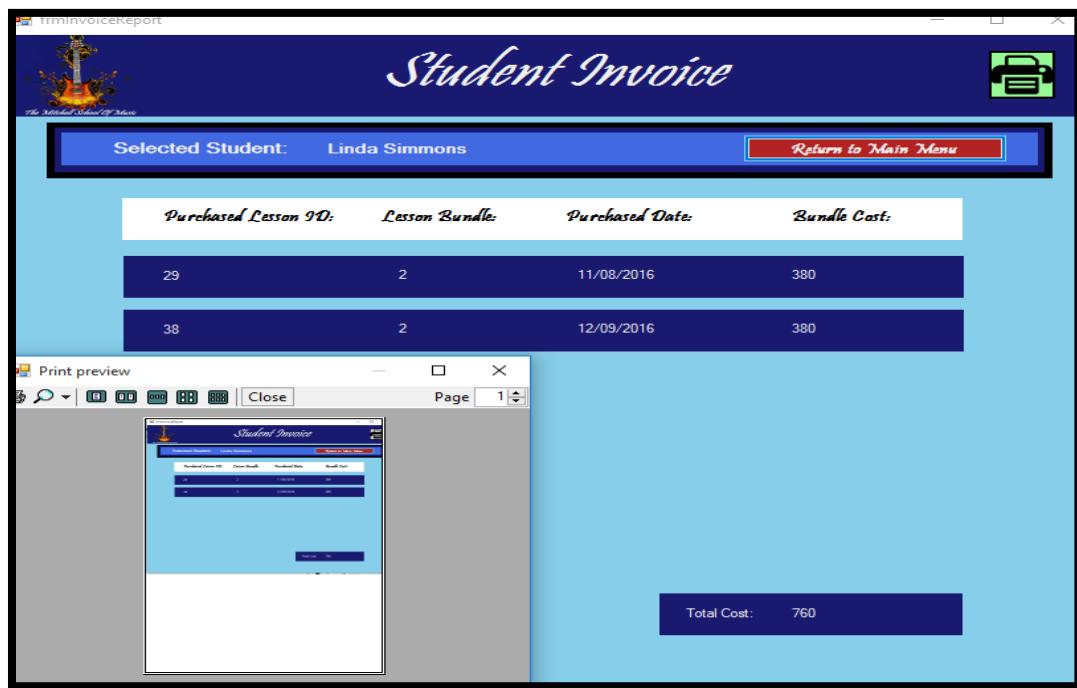
Process [B]) Form Navigation**2.20[A][1] – Return to Main Menu****2.20[B][1] Screenshot A**

The screenshot shows the same "frmInvoiceReport" application window as the previous one. The "Selected Student" is still "Linda Simmons". The main body of the window now displays a table of purchased lessons:

Purchased Lesson ID:	Lesson Bundle:	Purchased Date:	Bundle Cost:
29	2	11/08/2016	380
38	2	12/09/2016	380

At the bottom right, there is a dark blue footer bar with the text "Total Cost: 760".

2.20[B][1] Screenshot B**Process [A]) Print Operation****2.20[A][1] – Print Operation****2.20[C][1] Screenshot A**

2.20[C][1] Screenshot B

| Chapter 5 |

| Evaluation |

System Evaluation

System Purpose and Operations

The initial request for the development and implementation of this system was pressed by the owner of Mitchell's School of Music, Peter Mitchell. It was directly expressed that the primary and general purpose of this system was to provide both staff and management with the ability to easily and effectively store, access, and modify large quantities of information. Many steps were taken to ensure that the design and functionality of the system was able to support this intention, and to effectively meet the various requirements established. For the initial segmentation of this chapter, it has been deemed beneficial to address the various components of this system, along with their purpose and operations:

SQL Tables

Tables could easily be identified as one of the most important aspects of the database system. This is primarily for the reason that tables are responsible for the storage and thus management of all information entered into the system. This essentially means that tables are responsible for the formation of a base structure for the system, which all other components can access. For instance, queries will merely use the restrictions which are established by the user, and then proceed to access the tables, locating the desired information and then relay it back to the user. In this system, there were numerous tables established, one for each of the following; "Students", "Teachers", "Instruments", "Grades", "Lesson Bundles", "Purchased Lessons", "Scheduled Lessons", "Reallocated Students", "Archived Scheduled Lessons", "Lesson Dates", "Archived Lesson Dates".

However, the extensive importance of these components, means that extensive care and consideration must be taken to ensure that they are constructed in a beneficial and safe manner. On this note, it would prove beneficial to address the fact that there are a few modifications which can and have been made to the tables, and their fields directly, as to modify and limit the input of information.

Most importantly, we need to address the data types which are assigned to the fields. These are used to dictate the overall function of the selected field, and restrict the flow of information, depending on the values which are entered. For instance, a few of the most common data types used within this database were: NCHAR, NVCHAR, INT, BIT, and Date Time. The NVCHAR or NCHAR data types in most cases represent a string value. This means that they will accept values that contain essentially all characters (Number, Symbol,

Letters etc). INT would be used to restrict the user from entering any character other than a number into the selected field, and Date Time will ensure that the user enters a value that represents a realistic date value, enforcing a desired format. Finally, it would also be beneficial to address the BIT data type, which would dictate that the selected field can store one of two existing values: True (1) or False(0).

Additionally, to further personalise the selected data field, the user can dictate whether or not NULLs can exist. Essentially, this means that should NULL be allowed, the user possess the ability to create a new record and leave the affect field blank. These blank fields contain NULL values. In most cases all fields present within the database should be NOT NULL, however in the event that there is a field which may not be necessary, this may be beneficial.

Finally, I should address the concept of Primary Keys within these tables, For Primary Key is the term used to identify a field which possesses the responsibility of representing each record present within the table with a unique and commonly automatically assigned value. To achieve this, the user can simply select the desired field, and then set the "IsIdentify" property to true. Upon doing this, they will also gain access to additional properties, including those associated with the automatic assignment of unique values. From here the user may make a few modifications, including the incrementing value between each record ID.

On the note of primary keys, is important to address the overall concept of relationships, for primary keys are essential within the process to effectively connect multiple tables together, providing them with the ability to reference data between each other. This is one of the most beneficial aspects of a database system, as it provides users with the ability to create an extensively large system, while reducing if not totally removing the duplication of information. For instance, in a standard flat file system, should a student wish to schedule a class, the business would need to document information regarding the class, and the student each time a new record is made. In a relational database however, individual tables can be assigned to individual aspects, in this instance students and scheduled lessons, then a student ID can be passed into the scheduled lessons, acting as a foreign key. This will then create the connection, and thus the ability to reference information from the student record.

Queries

Queries could simply be identified as collections of SQL code which are used to complete specific operations or actions. In most cases, especially within this database system, queries would be used to search the database system and return desired values. On this note, it would prove beneficial to address, that various parameters can be established alongside these queries, as to further restrict the records which are being returned. For instance, "SELECT Students * WHERE Student ID = 2". This should mean that only information regarding the student in possession of the student ID 2 should be returned.

In addition to locating and presenting information from the system to the user, queries can also be assigned the task of taking information from the user and using it to either update or add information to the system. This would be done with the use of the keywords "UPDATE or ADD". These have been used on a few occasions through the present system, to automatically calculate an End Date value, and then returning it to the system for each scheduled lesson individually, or to add entirely new records to various tables throughout the system.

It would also prove beneficial to address the fact that when selecting data from the system with the use of a SELECT query, the user can implement an SQLDATAREADER to access the returned information. Throughout this database system, these variables would be used to store the information to local instances, which could have then been displayed, or modified accordingly.

Forms

Forms are extensively important components of a database system, and could be identified as the end product of the system, for they act as the interface of information between the user and the system. For instance, upon accessing the database, the user will be met with a splash-screen, which would simply act as an introduction to the database, and would simulate a loading process. From this point the user would then be presented a main menu which contains an assortment of buttons which provide simplistic access to all addition forms. The buttons used within forms are important to acknowledge, for they are one of the most common and simplistic methods to use when initialising various operations within the database. For instance, selecting the open form button will simply cause the associated form to display. More importantly, forms have been established around each of the tables present within the system, with the primary focus of providing users with the ability to easily locate, add, delete or modify information. To accomplish this effect, text boxes, list

boxes, combo boxes and check boxes can all be used to store relevant details, and provide the user with the ability to update them accordingly.

To display information from the system, specialised forms have been made for each of the tables present within the system. These forms contain many components, an example of which are list boxes, that in this instance are used to display the information for each record to the user. Accompanying these display components are labels that would be used as headers to identify the information being expressed. In some cases, DataGrid components have also been added to the forms to display record information. For instance, for scheduled lessons, list boxes will detail the selected student, and the DataGrid would be used to detail the scheduled lesson records. In addition to this there would be multiple buttons present within the form to provide users with the ability to effectively navigate between associated records, be this to view the next, previous, first, or last record. On the topic of navigation, there are also combo boxes which have been implemented into the forms with the intention of acting as search operations. For this reason, they will detail information associated with the selected record, and provide users with the ability to quickly access the desired records.

Finally, these reports will contain various buttons to link important and beneficial forms, examples of such being the Add form, deletion confirmation form, and a return button to access the Primary menu form. These forms either provide the functionality of navigation, information manipulation, to add records, or to delete records accordingly.

Password

A password is a function used to effectively limit access to a specific system or location by locking it. A password could simply be identified as a collection of alphabetical and numerical values which are entered in a specific order. This provides the developer or system manager with the ability to either establish a general password, or individual passwords for every member of staff. Overall, this would mean that accessing the system would cause an initial prompt to appear, requesting the user to enter the designated password. Should the user enter the correct password (One which matches the exact format and contents of that which was designated), the database will load, while entering an incorrect password will reject the user access to the system. For most systems, a password login will provide the user with three attempts, which when exceeded will lock the system or user, depending on the current systems properties.

Meeting the user Requirements:**1) *Splash Screen***

- A) Must initialise and load database system.
- B) Must provide loading progression graphic(s).
- C) Must provide text-based user feedback.
- D) Must link to Main Menu form.
- E) This must be the initial form to load.

The initial form for the system, that will appear upon loading the program is the splash screen. This form is relatively simple, displaying the business name and logo, accompanied with the more important loading bar component, and accompanied loading label. Code has been established so that upon loading the form a timer will initialise, and begin incrementing the loading bar at a steady pace, which is visually represented by the presence of a green bar that would proceed to travel along the component. In addition to this, in response to reaching designated milestones within the loading progress, the loading label will update to display three different messages, during one multiple operations may occur. Finally, upon reaching 100%, the Main Menu form will load.

2) *Main Menu form*

- A) Must provide a link to the private tuition Form.
- D) Must display potential links (Unused buttons) to other sections of the database.
- E) Must provide ability to exit program.

This is an extensively simple form, which is solely used to access the different areas of the database system. Currently however, the Private Tuition is the only completed and generally operational section of the system, meaning that this is the only button that will cause a new form to load upon selection. Attempting to select any of the other buttons present within the form will result in the display of an error message, stating that the select section is still under construction, and will instead point the user towards the selection of the "Private Tuition" button.

3) *Private Tuition Form*

- A) Must provide a link to Student Table.
- B) Must provide a link to Teacher Table.
- C) Must provide a link to Instrument Table.
- D) Must provide a link to Room Table.
- E) Must provide a link to Grade Table.
- F) Must provide a link to Lesson Bundle Table.
- G) Must provide a link to Scheduled Lessons Table.
- H) Must provide a link to Purchased Lessons Table.
- I) Must provide a link to Schedule calendar.
- J) Must provide a link to Login.
- K) Must provide a link back to Main Menu.
- L) Must present feedback regarding login status.

This form is used as a general navigational hub, providing users with the ability to select from a large array of buttons. Each of the buttons present within the form will be linked to a corresponding form, or operation.

4) *Student Table*

- A) Must display student information in an ordered and easy to understand fashion.
- B) Must be able to easily navigate between student records.
- C) Must be able to search for students.
- D) Must provide link to Add field form.
- E) Must provide ability to append details.
- F) Must provide a link to Login or Confirmation form depending if no user is logged in.
- G) Must provide a link back to Private tuition.

The overall purpose of this form, is to provide users with the ability to easily view, and access further forms which provide the ability to modify the information present within the table. To express the information, there are multiple list boxes present alongside one wall of the form, which is used to store and display information in association to the select student record. Then to navigate throughout the many records that can be present within the associated table, a few methods have been implemented. For standard navigation, there are four buttons present within this form. One when selected will update the form to display information associated with the previous student record, and the other the next.

The other two buttons provide the users with the ability to quickly and easily navigate to the first and last records. Finally, for a more direct form of navigation, the decision was made to implement a combo box within the form, which will detail information regarding each record present within the table, in this case the students full name. This provides users with the ability to effectively and easily locate the desired student record and view their information.

Moving on, there are a collection of buttons present within this form, labelled “Add”, “Update”, and “Remove”. Working in order, selecting the first button will simply cause the Add form to appear, while selecting the “Update” button will cause the same form to load, however this time the details associated with the select student will be brought across. Finally, upon selecting the Remove button, depending on whether the user is logged on, they will either be presented with the login form, or a confirmation form, that displays some simple details associated with the selected student record.

Finally, present at the end of the form is a simple button which would be used to return to the Private Tuition form.

5) *Teacher Table*

- A) Must display teacher information in an ordered and easy to understand fashion.
- B) Must be able to easily navigate between teacher records.
- C) Must be able to search for teachers.
- D) Must provide link to Add field form.
- E) Must provide ability to append details.
- F) Must provide a link to Login or Confirmation form depending if no user is logged in.
- G) Must provide a link back to Private tuition.

The overall purpose of this form, is to provide users with the ability to easily view, and access further forms which provide the ability to modify the information present within the table. To express the information, there are multiple list boxes present alongside one wall of the form, which is used to store and display information in association to the select Teacher record. Then to navigate throughout the many records that can be present within the associated table, a few methods have been implemented. For standard navigation, there are four buttons present within this form. One when selected will update the form to display information associated with the previous teacher record, and the other the next.

The other two buttons provide the users with the ability to quickly and easily navigate to the first and last records. Finally, for a more direct form of navigation, the decision was made to implement a combo box within the form, which will detail information regarding each record present within the table, in this case the teacher full name. This provides users with the ability to effectively and easily locate the desired teacher record and view their information.

Moving on, there are a collection of buttons present within this form, labelled “Add”, “Update”, and “Remove”. Working in order, selecting the first button will simply cause the Add form to appear, while selecting the “Update” button will cause the same form to load, however this time the details associated with the select teacher will be brought across. Finally, upon selecting the Remove button, depending on whether the user is logged on, they will either be presented with the login form, or a confirmation form, that displays some simple details associated with the selected student record.

Finally, present at the end of the form is a simple button which would be used to return to the Private Tuition form.

6) *Instrument Table*

- A) Must display instrument information in an ordered and easy to understand fashion.
- B) Must be able to easily navigate between instrument records.
- C) Must be able to search for instruments.
- D) Must provide link to Add field form.
- E) Must provide ability to append details.
- F) Must provide a link to Login or Confirmation form depending if no user is logged in.
- G) Must provide a link back to Private tuition.

The overall purpose of this form, is to provide users with the ability to easily view, and access further forms which provide the ability to modify the information present within the table. To express the information, there are multiple list boxes present alongside one wall of the form, which is used to store and display information in association to the select instrument record. Then to navigate throughout the many records that can be present within the associated table, a few methods have been implemented. For standard navigation, there are four buttons present within this form. One when selected will update

the form to display information associated with the previous student record, and the other the next. The other two buttons provide the users with the ability to quickly and easily navigate to the first and last records. Finally, for a more direct form of navigation, the decision was made to implement a combo box within the form, which will detail information regarding each record present within the table, in this case the information full name. This provides users with the ability to effectively and easily locate the desired instrument record and view their information.

Moving on, there are a collection of buttons present within this form, labelled “Add”, “Update”, and “Remove”. Working in order, selecting the first button will simply cause the Add form to appear, while selecting the “Update” button will cause the same form to load, however this time the details associated with the select information will be brought across. Finally, upon selecting the Remove button, depending on whether the user is logged on, they will either be presented with the login form, or a confirmation form, that displays some simple details associated with the selected instrument record.

Finally, present at the end of the form is a simple button which would be used to return to the Private Tuition form.

7) Room Table

- A) Must display Room information in an ordered and easy to understand fashion.
- B) Must be able to easily navigate between Room records.
- C) Must be able to search for Rooms.
- D) Must provide link to Add field form.
- E) Must provide ability to append details.
- F) Must provide a link to Login or Confirmation form depending if no user is logged in.
- G) Must provide a link back to Private tuition.

The overall purpose of this form, is to provide users with the ability to easily view, and access further forms which provide the ability to modify the information present within the table. To express the information, there are multiple list boxes present alongside one wall of the form, which is used to store and display information in association to the select Room record. Then to navigate throughout the many records that can be present within the associated table, a few methods have been implemented. For standard navigation, there are four buttons present within this form. One when selected will update the form to

display information associated with the previous student record, and the other the next. The other two buttons provide the users with the ability to quickly and easily navigate to the first and last records. Finally, for a more direct form of navigation, the decision was made to implement a combo box within the form, which will detail information regarding each record present within the table, in this case the information full name. This provides users with the ability to effectively and easily locate the desired Room record and view their information.

Moving on, there are a collection of buttons present within this form, labelled “Add”, “Update”, and “Remove”. Working in order, selecting the first button will simply cause the Add form to appear, while selecting the “Update” button will cause the same form to load, however this time the details associated with the select information will be brought across. Finally, upon selecting the Remove button, depending on whether the user is logged on, they will either be presented with the login form, or a confirmation form, that displays some simple details associated with the selected Room record.

Finally, present at the end of the form is a simple button which would be used to return to the Private Tuition form.

8) Grade Table

- A) Must display Grade information in an ordered and easy to understand fashion.
- B) Must be able to easily navigate between Grade records.
- C) Must be able to search for Grades.
- D) Must provide link to Add field form.
- E) Must provide ability to append details.
- F) Must provide a link to Login or Confirmation form depending if no user is logged in.
- G) Must provide a link back to Private tuition.

The overall purpose of this form, is to provide users with the ability to easily view, and access further forms which provide the ability to modify the information present within the table. To express the information, there are multiple list boxes present alongside one wall of the form, which is used to store and display information in association to the select Grade record. Then to navigate throughout the many records that can be present within the associated table, a few methods have been implemented. For standard navigation, there are four buttons present within this form. One when selected will update the form to

display information associated with the previous teacher record, and the other the next. The other two buttons provide the users with the ability to quickly and easily navigate to the first and last records. Finally, for a more direct form of navigation, the decision was made to implement a combo box within the form, which will detail information regarding each record present within the table, in this case the teacher full name. This provides users with the ability to effectively and easily locate the desired teacher record and view their information.

Moving on, there are a collection of buttons present within this form, labelled “Add”, “Update”, and “Remove”. Working in order, selecting the first button will simply cause the Add form to appear, while selecting the “Update” button will cause the same form to load, however this time the details associated with the select Grade will be brought across. Finally, upon selecting the Remove button, depending on whether the user is logged on, they will either be presented with the login form, or a confirmation form, that displays some simple details associated with the selected Grade record.

Finally, present at the end of the form is a simple button which would be used to return to the Private Tuition form.

9) Lesson Bundle Table

- A) Must display Lesson Bundle information in an ordered and easy to understand fashion.
- B) Must be able to easily navigate between Lesson Bundle records.
- C) Must be able to search for Lesson Bundle.
- D) Must provide link to Add field form.
- E) Must provide ability to append details.
- F) Must provide a link to Login or Confirmation form depending if no user is logged in.
- G) Must provide a link back to Private tuition.

The overall purpose of this form, is to provide users with the ability to easily view, and access further forms which provide the ability to modify the information present within the table. To express the information, there are multiple list boxes present alongside one wall of the form, which is used to store and display information in association to the select Lesson Bundle record. Then to navigate throughout the many records that can be present within the associated table, a few methods have been implemented. For standard navigation,

there are four buttons present within this form. One when selected will update the form to display information associated with the previous Lesson Bundle record, and the other the next. The other two buttons provide the users with the ability to quickly and easily navigate to the first and last records. Finally, for a more direct form of navigation, the decision was made to implement a combo box within the form, which will detail information regarding each record present within the table, in this case the Lesson Bundle title. This provides users with the ability to effectively and easily locate the desired Lesson Bundle record and view their information.

Moving on, there are a collection of buttons present within this form, labelled “Add”, “Update”, and “Remove”. Working in order, selecting the first button will simply cause the Add form to appear, while selecting the “Update” button will cause the same form to load, however this time the details associated with the select Lesson Bundle will be brought across. Finally, upon selecting the Remove button, depending on whether the user is logged on, they will either be presented with the login form, or a confirmation form, that displays some simple details associated with the selected student record.

Finally, present at the end of the form is a simple button which would be used to return to the Private Tuition form.

10) Scheduled Lesson Table

- A) Must display scheduled lessons information by student, in an order and easy to understand fashion.
- B) Must display associated details regarding: Student, Teacher and instrument.
- C) Must be able to easily navigate between scheduled lessons records.
- D) Must be able to search for Student.
- E) Must provide link to Add field form.
- F) Must provide ability to append details.
- G) Must provide a link to Login or Confirmation form depending if no user is logged in.
- H) Must provide a link back to Private tuition.

The overall purpose of this form is to display information regarding Scheduled Lesson Records, and to provide users with easy access to associated data modification forms. To display the information, there are a multiple list boxes present within the form, which are categorised into three groups, one associated with student information, Grade information,

and Teacher Information. Accompanying these components are a collection of labels that are used as headers, to help users further understand the information being displayed. Along the lower section of the form, there is a Data Grid which is used to detail information regarding all Scheduled lessons associated with the selected student record.

Present along the opposite side of the form is a collection of buttons, the upper collection of which are used to provide the user with the ability to effectively navigate between student records present within the system. There are four basic navigational buttons, one which is used to update the form display to express information regarding the previous student, another to view the next student, and the final two provide the ability to quickly access the first and last student records. A more advance and effective navigational tool also present within this area is a simple combo box, that will list student information, and essentially provide users with the ability to locate a desired student, and then view their associated details.

Moving on, there are a collection of buttons present within this form, labelled "Add", "Update", and "Remove". Working in order, selecting the first button will simply cause the Add form to appear, while selecting the "Update" button will cause the same form to load, however this time the details associated with the select Schedule Lesson will be brought across, however should no value be present, an error message will appear stating that the user must select a record. Finally, upon selecting the Remove button, depending on whether the user is logged on, they will either be presented with the login form, or a confirmation form, that displays some simple details associated with the selected Scheduled Lesson record.

Finally, present at the end of the form is a simple button which would be used to return to the Private Tuition form.

11) Purchased Lesson Table

- A) Must display Purchased lessons information by student, in an order and easy to understand fashion.
- B) Must display associated details regarding: Student.
- C) Must be able to easily navigate between Purchased lessons records.
- D) Must be able to search for Student.
- E) Must provide link to Add field form.
- F) Must provide ability to append details.

- G) Must provide a link to Login or Confirmation form depending if no user is logged in.
- H) Must provide a link back to Private tuition.

The overall purpose of this form is to display information regarding Purchased Lesson Records, and to provide users with easy access to data modification forms. To display the information, there are a multiple list boxes present within the form, which are categorised into three groups, one associated with student information, Grade information, and Lesson Bundle Information. Accompanying these components are a collection of labels that are used as headers, to help users further understand the information being displayed. Along the lower section of the form, there is a Data Grid which is used to detail information regarding all purchased lessons associated with the selected student record.

Present along the opposite side of the form is a collection of buttons, the upper collection of which are used to provide the user with the ability to effectively navigate between student records present within the system. There are four basic navigational buttons, one which is used to update the form display to express information regarding the previous student, another to view the next student, and the final two provide the ability to quickly access the first and last student records. A more advance and effective navigational tool also present within this area is a simple combo box, that will list student information, and essentially provide users with the ability to locate a desired student, and then view their associated details.

Moving on, there are a collection of buttons present within this form, labelled “Add”, “Update”, and “Remove”. Working in order, selecting the first button will simply cause the Add form to appear, while selecting the “Update” button will cause the same form to load, however this time the details associated with the select Purchased Lesson will be brought across, however should no value be present, an error message will appear stating that the user must select a record. Finally, upon selecting the Remove button, depending on whether the user is logged on, they will either be presented with the login form, or a confirmation form, that displays some simple details associated with the selected purchased Lesson record.

Finally, present at the end of the form is a simple button which would be used to return to the Private Tuition form.

12) Schedule Calendar

- A) Must provide user with the ability to select a specific date.
- B) Must provide access to calendar times.
- C) Must provide link back to Private tuition.

This form is the first calendar form which the user will be able to access, and would be used to identify the desired day for the calendar search operation. For this reason, the main component is a calendar that is present within the centre of the form. This provides users with the ability to easily locate the desired date and select it. It is important to address however that in the event the user selects a weekend date or a summer date, error messages will appear stating that they are invalid. Upon selecting a valid date, the next calendar form regarding time selection will appear.

Additional components present within this form are a combo box that will detail all possible months, and will provide users with the ability to easily and quickly search through the calendar. Finally, located within the lower section of the form is a simple return button that provides users with the ability to access the Private Tuition Form.

13) Calendar Time

- A) Must provide user with the ability to select a specific time.
- B) Must provide access to Scheduled Classes.
- C) Must provide access back to schedule calendar dates.

This form provides users with the ability to select the desired time range which would be used within the scheduled lesson search operation. For this reason, the centre piece of this form is composed of a 4x4 grid of buttons, each of which displays a time value in intervals of 30 minutes. Upon selecting one of these buttons, the final calendar form will appear, this being the scheduled classes.

Other components present within this form is a label present within the upper section which will simply represent the previously selected date, and a button present within the lower section of the form which provides the user with the ability to return to the previous calendar form.

14) Scheduled Classes

- A) Must display classes which have been established for the specific time and date selected.
- B) Must provide a link to scheduled lessons table.
- C) Must provide link to Private Tuition.
- D) Must provide link back to calendar times.

The overall purpose of this form is to provide users with the ability to identify and reference scheduled lesson information in association to a specific date and time. For this reason, the centre piece of this form is composed of a collection of labels and picture boxes that make up custom information displays. In addition to this, there are a few labels present within the upper-section of the form, and are used to ensure that user is aware of the current date and time which they are viewing information from.

Upon accessing this form, the system will automatically review all scheduled lessons present within the system, and identify those that belong to the selected time and date. Then they will be placed within the displays accordingly. It is also beneficial to address that fact the users possess the ability to quickly view extended collections of information in association to a scheduled lesson. This can be done by simply selecting the associated display, which will then cause the scheduled lessons form to appear, and will set the display to the associated ID.

Located along the lower end of the form are three individual buttons; “Return to Calendar”, “Schedule New Lessons”, and “Return to Main Menu”. The first button will complete the expected action, selecting it will cause the previous form of Calendar Times to appear. Selecting “Schedule New Lesson” will cause the add field form to appear, under the content that a new scheduled lesson will be added. Then finally, the “Return to Menu” button can be used to simply load the Private Tuition Form.

15) Login

- A) Must provide admins with the ability to login.
- B) Must prevent unauthorised

This form is simple, providing users with the ability to login in the event that they have previously been provided with login details. For this reason, this form simply consists of two textboxes, and two buttons. The first textbox is used to store the username value, and the second is password value. Finally, the buttons present within this form are login, and return.

Upon selecting the login button, the entered values will be validated, and if the details match, then the user will login, otherwise an error message will appear.

16) Add Field

- A) Must provide the user with the ability to easily enter new details (If logged in).
- B) Must validate entered data.
- C) Must Append Data to table if valid.
- D) Must Provide Extensive user feedback, to simplify system.
- E) Must contain a link to previously used form.

17) Confirmation

- A) Must display a confirmation option (If user logged in).
- B) Must provide a link to previously used form.

This sole purpose of this form is to act a final confirmation submitted by the user in relation to the deletion process of a selected record. This form will be extensively accessible, in the sense that all table-based forms will have an associated button. Upon selecting this button, the information associated with the selected record will be transferred and displayed within the confirmation form, as long as an admin is logged in.

This form provides the user with two possible buttons for them to select from, "Confirm" or "Cancel". The confirm button would be used to essentially finalise the deletion process and remove the selected record from the system. Alternatively, selecting the Cancel button will stop the operation from finalising, and will return the user to the previous form.

18) Upcoming lessons

- A) User must be able to view accurate and up-to-date information regarding the upcoming scheduled lessons.
- B) Users must be able to search for student or Teacher entities.
- C) User must be able to easily navigate through the displayed records.
- D) User must be able to navigate to Main Menu from this form.

The overall purpose of this form is to provide users with the ability to easily and effectively locate schedule information in association with specific students. For this reason, the centre

piece of this form consists of 6 individual displays, along with a prominent tool bar along the upper section, and a page count along the bottom (if needed).

It is important to initially address that the form toolbar contains a button that can be selected to change between teacher and student, along with an associated search bar. Should the button be set to teacher, the combo box will update to display information associated with each teacher record present within the system, alternatively should the button be set to student, then the combo box will display information associated with each student record. This will provide the user with the ability to easily select the desired student record, and then complete the search operation by selecting the associated button.

Once the search operation has been initialised, a validation check will be completed behind the scene to locate records that match the chosen information. This information will then be displayed within the various displays. It is also important to address that in the standard display can only support 6 records at one time. For this reason, should more than 6 records be identified, the user will be provided with the ability to access page navigational buttons along the lower end of the form, which can be used to view all information.

Finally, should the user wish to leave this form, an exit button is present within the form, providing the ability to return to the Private Tuition Form.

19) General

- A) System must be professional.
- B) System must be user friendly.
- C) System must be operational.

Overall extensive time and consideration was taken to ensure that this system was developed in a manner which met all of the mentioned requirements and more. For instance, the designed had been modified to make important information more prominent, buttons colour coded and therefore easier to recognise, and a consistent colour scheme to make the overall system appear more professional and organised. I feel that all of these points are important for the reason that they can improve the overall user experience in a beneficial manner.

In addition to this, I wanted to ensure that this system was actually operational. For this reason, a number of relatively complex operations where implemented, as to ensure that;

specific records were archived and thus removed from their origin table, specific fields were automatically calculated and re-entered into their necessary table.

How has data modelling improved system

There were various data modelling methods which were used through this document, and the general development process of the desired system. This was due to the fact that most provided the ability to construct an accurate plan regarding the systems contents and structure, while others outlined the development process, as to ensure that various deadlines were met.

To elaborate further, the project plan was used to structurally address the many steps which would be needed to professionally develop the system within a specific time period. This meant that the steps would be accompanied by a start and end date, specifying the amount of time which I could expend within a specific section. To accompany this, a GANNT chart was also developed, for this takes the information present within the project plan, and displays it in a more visually appealing and interruptible format. Following these plans, I was able to ensure that the system was finalised in time to meet the final deadline.

In addition, I also developed and used Data flow diagrams, Entity Relation Diagrams. These were important for they directly identified the various pieces of content which needed to be present within the system overall, and the connections which needed to exist between the included objects. These diagrams provided the ability to visually express these details, and thus provide a basic guideline for the construction of further planning of the system.

Review of end user feedback

Upon finalising the system, and releasing a version to the school for additional testing, a few individuals were provided with the ability to use and review the product. I then developed specialised questionnaires, which would be released to these individuals upon the completion of these trials, as to acquire potentially beneficial feedback.

The overall feedback from these questionnaires was astronomically positive. Most areas and functions of the system were regarded as extensively beneficial and operational. This helps enforce the fact that the system was developed to a standard in which it met all necessary requirements and functions necessary to accommodate the school and their operations. One recommendation which was made however relates to Mail Merging. This operation would be used to effectively transfer information from a database system to another document. Most commonly, this would be used within invoice or newsletter distribution.

Future recommendations

Having complete the system, it is important to note that I have identified a few features and suggestions which could prove beneficial to the school and the system in the future. This includes:

- ❖ Staff Members – While the system has been designed in a manner which accommodates users with little-to-no IT knowledge, it would still prove extremely beneficial to eventually establish staff training days, and issue associated learning materials. This is because the system will only be as effective as the individual using it, meaning that expanding the user's knowledge, might provide them with the ability to complete various tasks and operations at accelerated speeds and efficiency.
- ❖ Development process – While I feel that I was capable to meet the established requirements, and create a functional and effective system, it is important to identify the short time-frame, which was built around the planning and development process. This is due to the fact that I feel with extra time to further manipulate the system, additional features and operations could be implemented. It could prove beneficial to provide myself or another with an extended deadline to review the site and improve upon it at some point within the future.
- ❖ Passwords – While I have implemented a standard password into the site, which will ensure that only members of staff and management can access the database. It is still important to identify the fact that the school could benefit greatly by assigning each member of staff an individualise password or login details. This is due to the fact that not all members of staff should be able to complete various operations; such as delete or add a student. These processes would be completed by the school's secretary or management, and thus shouldn't even be accessible to some.
- ❖ Scheduled Classes Calendar – The calendar system present within the system will currently reference information from the scheduled lessons table, and while this is functional, it is not optimal in regards to efficiency. I would personally recommend, and would have generally liked to have updated the system to reference information from the lesson dates table instead.

- ❖ Mail Merge – One final recommendation which I would like to address towards the management of the school relates to Mail Merges. This is a simple practice used to effectively detail the information present within the database system, into a MS Word document. This could prove extensively beneficial within the creation of invoice documents.
- ❖ Record Deletion – I have recently reviewed the process and concept of record deletion within the system, and have come to the conclusion that on a large scale, it may prove relatively ineffective. Using a student record for reference, it is initially important to address the fact that an individual student ID can be associated with purchased lessons and scheduled lessons. This is problematic for when deleting a student record, their information will be removed from the origin table, yet records which access the now delete ID remain within the system.

Essentially, this could cause a build-up of scheduled and purchased records which have no connection to any student present within the system. For this reason, it may prove beneficial to update the system, and establish that when deleting a specific student record a check should be completed, as to ensure that records associated with the deleted record are also removed.

- ❖ Upcoming Lesson Filter – One of the features which I would like to have presented within the Upcoming Lesson form is a filter system. The initial idea was the users would be provided with the ability to sort the Schedule IDs by Date, Time etc., which would then elaborate into (Today, This week, This month or 13:00, 13:30, 14:00 etc.). This however fell through for various time related reasons, however I would still deem that this would be beneficial overall for the school to consider and potentially implement.

Personal Performance

Generally, I am pleased with the overall outcome of the developed system, and feel that I can associate some success with the moderate knowledge I have acquired in relations to database systems and similar programs. It is interesting to note however, that through this process, I have come face-to-face with general barriers that have never been met before., which encouraged some interesting problem solving. In addition to this, there were also numerous instances where I was requested to develop and operate aspects of a

database which I have never handled before, thus overall I feel that this process has strengthened by understanding of database systems, and has provided myself with the ability to operate more efficiently within future projects.

While designing and developing the system, the user requirements played an extremely important role, and were therefore referenced consistently. This was because I wanted to ensure that this system possessed all features requested by the owner, along with additional developments, as to establish an effective and professional program for the school. It is important to acknowledge the fact that the primary purpose of this system is to provide the school with the ability to easily and effectively store, retrieve and modify information from numerous aspects of the school. This meant that it was extremely important to properly analyse the school's operations, as to establish the necessary tables, queries and other functions. This was to ensure that all information and operations implemented into the system were accurate and beneficial to the school.

As previously mentioned, the user requirements were particularly important within this development process, thus direct and consistent communication was deemed necessary between myself and management. This process was used to establish a guideline regarding the development of the system, and the desired contents. This provided myself with the ability to augment the system accordingly. In addition to this, multiple deadlines were established in relation to various aspects of the database system. This was beneficial as it provided management with a consistent representation of the system's progress, and the ability to provide myself with updated requirements and recommendations. These deadlines were also important for the fact that they actively ensured that the database would be operational before the final date, and therefore the start of the new term. This final deadline was important to meet, for missing it could negatively impact the school, and my personal reputation.

A few of the most significant requirements revolved around making the system user friendly, professional, and easy to navigate. To accomplish this, the system design was frequently remodelled and updated to establish the most effective design which compliments the functionality of the system.

Strengths**❖ Time management**

Having completed the general review of my personal progress and performance throughout this project, I can confidently identify one of my primary strengths as time management. As has been previously mentioned within this section of the document, the systems requester established numerous deadlines through the production process, as to actively monitor and encourage progress. To ensure that these deadlines were met, each section of the system was planned accordingly, and provided with the necessary work application as to implement all necessary contents and more. As has been previously mentioned, this was extremely important to me, for missing deadlines could poorly represent my work, and professionalism.

❖ Design

Overall, it is important to address the fact that I am extremely pleased with the design and layout which has been implemented throughout the database. I feel that I have selected a colour scheme and general design which actively presents the system as organised, professional and user friendly. For instance, a standard header containing the school name, and logo was designed and applied to each form and report, as to create a consistent and professional appearance.

❖ Information Display

As a continuation and elaboration of the previous point, I am also pleased with the overall display of information. Having identified the overall importance of information within this system, a strong emphasis was built around the creation of a layout to accurately and effectively present the information. This was accomplished with the use of

❖ Implementation of effective operations

In addition to the design, another example of an extremely important feature which was evident within the database system related to operations and general procedures. It was important to ensure that the database system provided users with the ability to automatically complete various operations regarding the management of information within the school. I personally feel that the current

operations present within the database system are effective and complex in some cases.

❖ **Employer Interactions**

As has been previously mentioned throughout the document, it was extremely important to ensure that the system was developed in a manner which made it effective within the process of meeting requirements set by the employer. To ensure that this was the case, numerous deadlines were established to consistently express the current state of the system, and the individual aspects included. I have deemed this an advantage, for these opportunities provided the myself and the employer with the ability to communicate information, and for them to express any concerns or recommendations which they may possess upon viewing the system. This effectively ensured that the employer had full control over the outcome of the system and its operations.

Weaknesses

❖ **Work Load**

While I feel that I have properly handled and met the various deadlines that were established throughout this project, I must admit that the total workload seemed overwhelming at times. I feel that this might be associated to the fact that I may have over complicated some operations and scenarios, thus adding to my total work load. This was not preferable, for I would not wish to run the overall risk of rushing various aspects of the system, as to simply meet the deadlines. In this scenario the overall product would be quantity over quality.

❖ **Breaks throughout the system**

Throughout the development process of this system, there were numerous instances where developed forms, tables and forms etc. would stop operating as intended, in some cases breaking all together. This was likely caused to personal errors, however another possibility could relate to a mere lack of knowledge.

❖ **Employer feedback**

This weakness was quite tame throughout the development process, however it is important to address nevertheless. As previously mentioned, employer feedback is an extremely important detail to be collected during this development process, for

it provides control over the system. However, this also means that should the employer desire, entire areas of the system may need to be reworked or expanded upon. This could prove problematic in relations to the deadlines, for this would impose more work to be completed.

Improvements

❖ Workload

As previously mentioned, I believe that I handled the associated deadlines well, presenting examples for each completed process on time. I will admit however, the amount of workload associated with the development of this system seemed overwhelming at times. There are many reasons for this, however, I would predominantly associate this with over complication on my part. There were instances throughout this document, where I may have expanded upon points excessively, or became too thorough in relations to testing and implementing the site. I must state however that I do not regret this over extension of the project, for I believe that if I am going to complete a project, I am going to ensure that it is completed correctly and completely.

❖ System improvements

It is essential to address, that while I was able to finalise the system to a standard which I am pleased with, and one which meets the specified requirements, I have pondered potential improvements which could have been made. For instance, while reviewing the operations of the system, I noticed a few oversights within the schedule table. Currently, the user will document the lessons start date, and associated information, to which an end date will be provided. The user can then select whether the student had attended. In theory however this does not make sense, for the record in question represents a range of lessons, instead of individual instances. To be fair, I personally have not been able to construct a reasonable and plausible method to rectify this oversight that abides by the restrictions imposed by the ACCESS program.

Additionally, a relatively small change which I believed could and likely should be made also relates to the schedule table, along with the purchased lessons table. In both objects, the lesson bundles are individually assigned to records. This is ineffective for it means to track student lessons, both values must be calculated individually, which can result in negative values. Alternatively, I should have

directly connected the purchased lessons to the scheduled lessons table, for this would mean that a student would be required to purchase a lesson before making a schedule. This would help within the reduction of redundant information, and strengthen the system structure overall.

Finally, it should simply be addressed that there are likely small beneficial pieces of content which could improve the system overall if implemented. This could include new queries, tables, reports etc.

❖ **Employer Interactions**

I actively strived to provide the employer with as much control over the system as possible, however there were instances where I was required to operate on initiative and individually. This worked in the end, however it could have created a scenario in which the employer may not be pleased with the outcome of my product, and decide that it would need to be modified or scrapped. An example of this was in during the development of the reports and forms, which could have benefited from having the employer present.

❖ **Development program**

One of the largest problems which became apparent within the later stages of development regarded time and the associated strain of deadlines. While likely associated to relatively poor personal management, and an assortment of additional tasks, it is still extensively beneficial and important to address that fact that there a large number of features which I would have like to implement.

For these reasons however, a few corners were regrettably cut around the system. For instance, due to the overall constraint some lengthy functions such as the Add or Update for each individual table where essentially skipped. This was due to the fact that I had already created a functioning in association to the student table, and deemed it to be more beneficial to begin working on other areas, than to duplicate code and functions within the system.

Overall however, I would like to address that within an extended deadline, I would have loved to continue the development of this system, as to ensure that all of these functions where completed accordingly.

Appendices

Appendix 1

| System Questionnaires |

Initial Questionnaires:**Initial Questionnaire 1**

Jacobs Musical Teachings
| Company Questionnaire |



Name: Position:

Does your position require that you access and store information regarding the school, and its students on a regular bases? If Yes, approximately how often?

Yes No
Multiple instances a day.

Have you personally experience any limitations regarding the completion of tasks, due to the current information management system present within the school? If Yes, could you please elaborate on the specific problems.

Yes No
I occasionally find it difficult and time consuming to locate, and document files. Here and there, I will locate multiple instances of the same file, each containing contradictory information.

The prospect of implementing a new database system to correct these limitations has been considered. Would you be opposed to the development and establishment of a new system? If yes, could you please provide your reasoning.

Yes No

Do you currently possess any IT knowledge, regarding the management and procedural use of a database system? If yes, please feel free to elaborate.

Yes No
Very little, I attended classes within secondary school.

Jacobs Musical Teachings

| Company Questionnaire |



What do you think the system should be able to do? (Please relate to displayed content or functions)

- | | |
|--|--|
| <input type="checkbox"/> Organise and display upcoming scheduled lessons for teachers. | <input checked="" type="checkbox"/> Simplify student reallocation |
| <input type="checkbox"/> Display Student Lesson Details | <input checked="" type="checkbox"/> Simplify the allocation of Lesson invoices |
| <input type="checkbox"/> Automatically calculate Lesson costs | <input checked="" type="checkbox"/> Automatically calculate Lesson End Dates |
| <input type="checkbox"/> Other(s): | <input type="text"/> |

Could you define important requirements for this system?

- | | |
|---|---|
| <input checked="" type="checkbox"/> Must be User Friendly | <input checked="" type="checkbox"/> Must be Organised |
| <input type="checkbox"/> Must be Colourful | <input type="checkbox"/> Must be Simplistic |
| <input checked="" type="checkbox"/> Must be Professional | <input checked="" type="checkbox"/> Must be secure |
| <input type="checkbox"/> Other(s): | <input type="text"/> |

Signature: A. Rydor

P. Jacobson

Date: 12/01/2017

Initial Questionnaire 2

Jacobs Musical Teachings
[Company Questionnaire]



Name:

Position:

Does your position require that you access and store information regarding the school, and its students on a regular bases? If Yes, approximately how often?

Yes No

Have you personally experience any limitations regarding the completion of tasks, due to the current information management system present within the school? If Yes, could you please elaborate on the specific problems.

Yes No

The prospect of implementing a new database system to correct these limitations has been considered. Would you be opposed to the development and establishment of a new system? If yes, could you please provide your reasoning.

Yes No

Do you currently possess any IT knowledge, regarding the management and procedural use of a database system? If yes, please feel free to elaborate.

Yes No

Jacobs Musical Teachings

| Company Questionnaire |



What do you think the system should be able to do? (Please relate to displayed content or functions)

- Organise and display upcoming scheduled lessons for teachers. Simplify student reallocation
- Display Student Lesson Details Simplify the allocation of Lesson invoices
- Automatically calculate Lesson costs Automatically calculate Lesson End Dates
- Other(s): *Simplify the process of recording student attendance.*

Could you define important requirements for this system?

- Must be User Friendly Must be Organised
- Must be Colourful Must be Simplistic
- Must be Professional Must be secure
- Other(s):

Signature: P. Scorch

Date: 12/01/2017

P. Jacobson

Initial Questionnaire 3

Jacobs Musical Teachings

| Company Questionnaire |



Name:

Position:

Does your position require that you access and store information regarding the school, and its students on a regular bases? If Yes, approximately how often?

Yes No

Occasionally.

Have you personally experience any limitations regarding the completion of tasks, due to the current information management system present within the school? If Yes, could you please elaborate on the specific problems.

Yes No

There was an instance in which I was scheduled for two classes at the same time.

The prospect of implementing a new database system to correct these limitations has been considered. Would you be opposed to the development and establishment of a new system? If yes, could you please provide your reasoning.

Yes No

Do you currently possess any IT knowledge, regarding the management and procedural use of a database system? If yes, please feel free to elaborate.

Yes No

I have partaken in high level IT classes, therefore I feel relatively adept.

Jacobs Musical Teachings

| Company Questionnaire |



What do you think the system should be able to do? (Please relate to displayed content or functions)

- | | |
|---|---|
| <input checked="" type="checkbox"/> Organise and display upcoming scheduled lessons for teachers. | <input checked="" type="checkbox"/> Simplify student reallocation |
| <input checked="" type="checkbox"/> Display Student Lesson Details | <input type="checkbox"/> Simplify the allocation of Lesson invoices |
| <input type="checkbox"/> Automatically calculate Lesson costs | <input type="checkbox"/> Automatically calculate Lesson End Dates |

Other(s):

Could you define important requirements for this system?

- | | |
|---|---|
| <input checked="" type="checkbox"/> Must be User Friendly | <input checked="" type="checkbox"/> Must be Organised |
| <input type="checkbox"/> Must be Colourful | <input type="checkbox"/> Must be Simplistic |
| <input type="checkbox"/> Must be Professional | <input checked="" type="checkbox"/> Must be secure |

Other(s): *The system should be informative, and accurate.*

Signature: Jacob Trainer

Date: 21/01/2017

P. Jacobson

Final Questionnaires:**Final Questionnaire 1**

Jacobs Musical Teachings
| Company Questionnaire |



Name: **Alexa Rydor**

Relation to business: **Secretary**

Thank you for using the system!

Could you now provide feedback about your experience with the system, by completing the following rating system?

Rank Keys:
1 - Poor,
5 - Average,
10 - Excellent.

1	2	3	4	5	6	7	X	9	10
---	---	---	---	---	---	---	----------	---	----

How user friendly was the system?

1	2	3	4	5	6	7	X	9	10
---	---	---	---	---	---	---	----------	---	----

How beneficial was the system?

1	2	3	4	5	6	7	X	9	10
---	---	---	---	---	---	---	----------	---	----

How organised was the system?

1	2	3	4	5	6	X	8	9	10
---	---	---	---	---	---	----------	---	---	----

How visually appealing was the system?

1	2	3	4	5	6	X	8	9	10
---	---	---	---	---	---	----------	---	---	----

How easy was it to navigate the system?

1	2	3	4	5	6	X	8	9	10
---	---	---	---	---	---	----------	---	---	----

How difficult was it to understand the system?

1	2	3	4	5	6	X	8	9	10
---	---	---	---	---	---	----------	---	---	----

How informative was the content?

1	2	3	4	5	6	X	8	9	10
---	---	---	---	---	---	----------	---	---	----

How would you rate this system overall?

1	2	3	4	5	6	X	8	9	10
---	---	---	---	---	---	----------	---	---	----

How would you personally improve the System?

I am extensively pleased with the outcome of this system, however if I had to make one recommendation, I would ask to include the ability to mail merge information. This would make it much easier to distribute important pieces of documentation and the included information.

Signature: **A. Rydor**

Date: **12/01/2017**

P. Jacobson

Final Questionnaire 2

Jacobs Musical Teachings

| Company Questionnaire |



Name:

Patrick Scorch

Relation to business:

Teacher

Thank you for using the system!

Could you now provide feedback about your experience with the system, by completing the following rating system?

Rank Keys:

1 - Poor,
5 - Average,
10 - Excellent.

How user friendly was the system?

1	2	3	4	5	6	7	X	9	10
---	---	---	---	---	---	---	---	---	----

How easy was it to navigate the system?

1	2	3	4	5	6	7	X	9	10
---	---	---	---	---	---	---	---	---	----

How beneficial was the system?

1	2	3	4	5	6	7	X	9	10
---	---	---	---	---	---	---	---	---	----

How easy was it to understand the system?

1	2	3	4	5	6	7	X	9	10
---	---	---	---	---	---	---	---	---	----

How organised was the system?

1	2	3	4	5	6	7	X	9	10
---	---	---	---	---	---	---	---	---	----

How informative was the content?

1	2	3	4	5	6	7	X	9	10
---	---	---	---	---	---	---	---	---	----

How visually appealing was the system?

1	2	3	4	5	6	X	8	9	10
---	---	---	---	---	---	---	---	---	----

How would you rate this system overall?

1	2	3	4	5	6	7	X	9	10
---	---	---	---	---	---	---	---	---	----

How would you personally improve the System?

Signature: P. Scorch

P. Jacobson

Date: 12/01/2017

Final Questionnaire 3

Jacobs Musical Teachings

| Company Questionnaire |

Name: Relation to business: Thank you for using the system!

Could you now provide feedback about your experience with the system, by completing the following rating system?

Rank Keys:

1 - Poor,
5 - Average,
10 - Excellent.

How user friendly was the system?

1	2	3	4	5	6	7	8	X	10
---	---	---	---	---	---	---	---	---	----

How easy was it to navigate the system?

1	2	3	4	5	6	7	X	9	10
---	---	---	---	---	---	---	---	---	----

How beneficial was the system?

1	2	3	4	5	6	7	X	9	10
---	---	---	---	---	---	---	---	---	----

How difficult was it to understand the system?

1	2	3	4	5	6	7	8	9	X
---	---	---	---	---	---	---	---	---	---

How organised was the system?

1	2	3	4	5	6	7	X	9	10
---	---	---	---	---	---	---	---	---	----

How informative was the content?

1	2	3	4	5	6	7	X	9	10
---	---	---	---	---	---	---	---	---	----

How visually appealing was the system?

1	2	3	4	5	6	7	X	9	10
---	---	---	---	---	---	---	---	---	----

How would you rate this system overall?

1	2	3	4	5	6	7	8	X	10
---	---	---	---	---	---	---	---	---	----

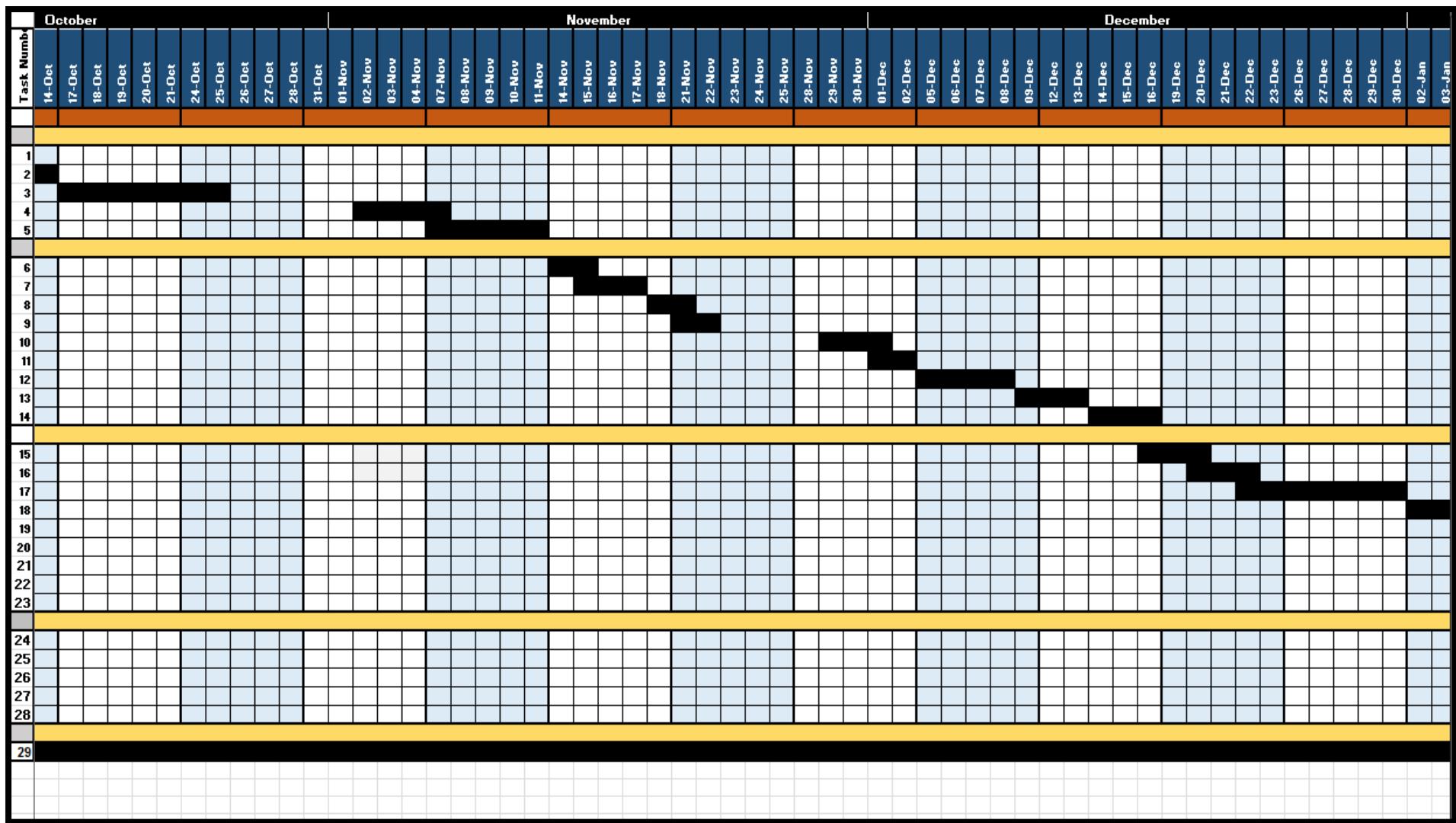
How would you personally improve the System?

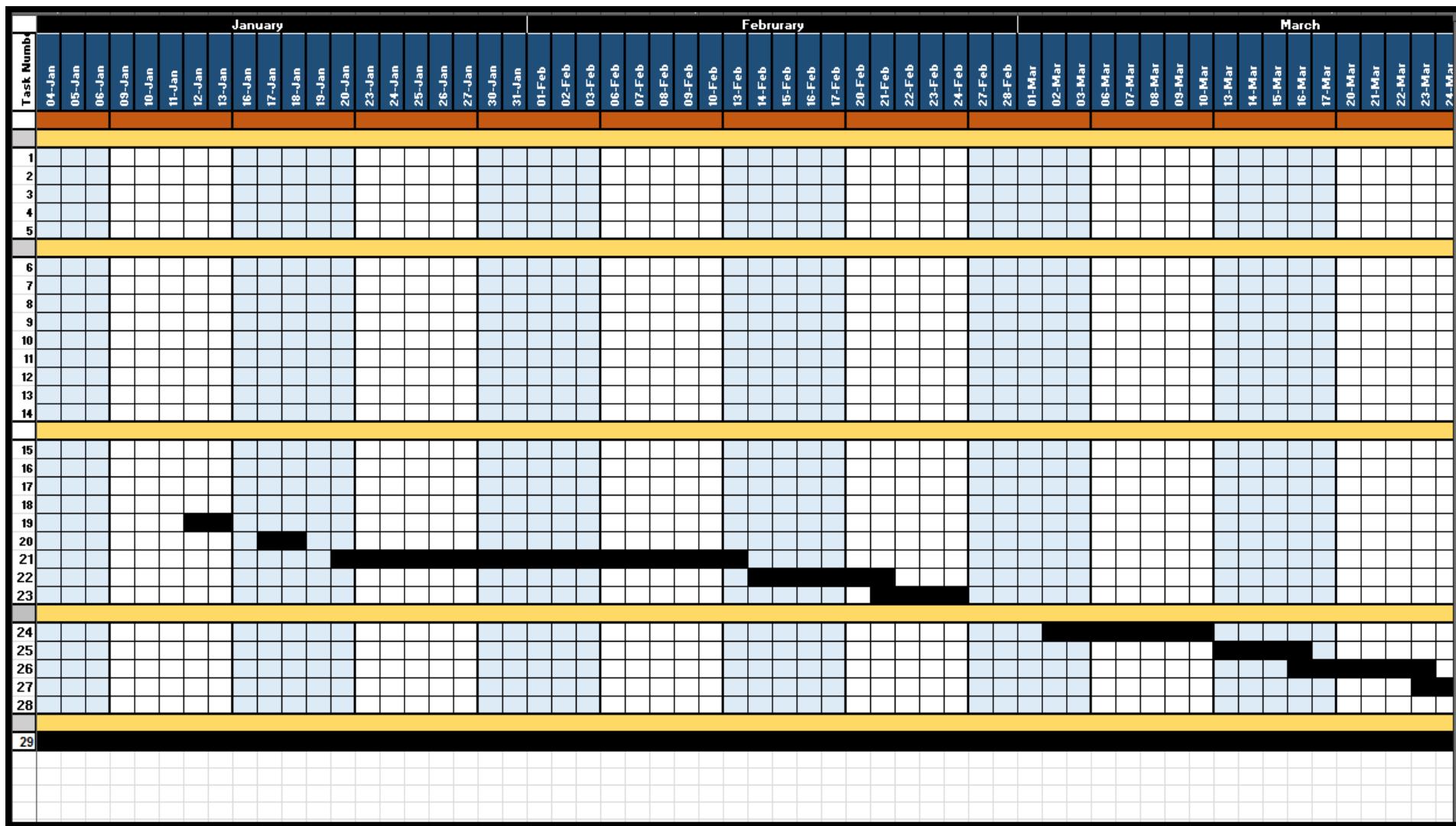
--	--	--	--	--	--	--	--	--	--

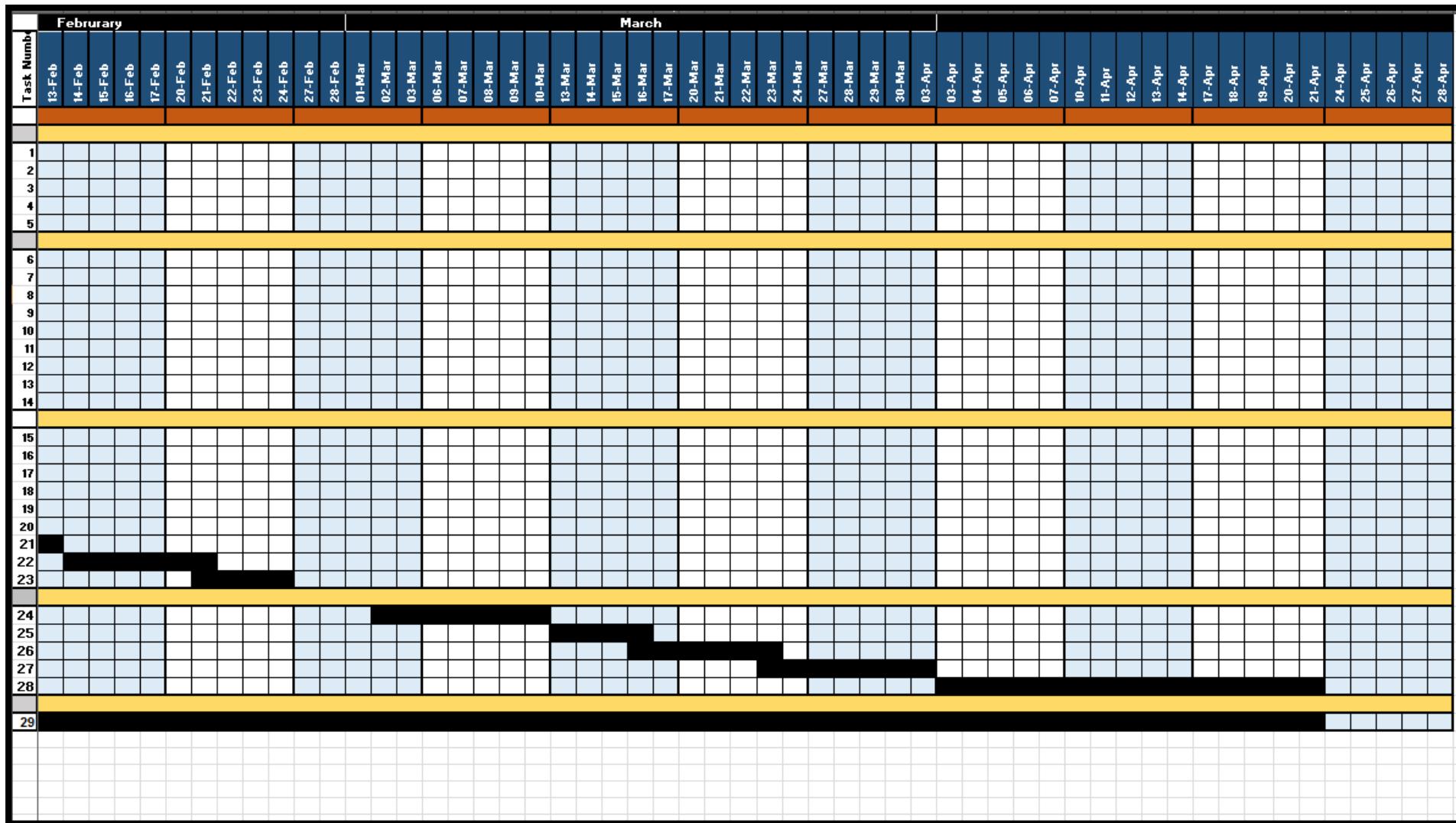
Signature: Jacob-Trainer*P. Jacobson*Date 21/01/2017

Appendix 2

| GANNT Chart |







Appendix 3

| ER Diagram |

What is an ER Diagram?

An ER diagram could be used to graphically represent key pieces of information present within the system, expressing the relations between data entities and their associated attributes. To elaborate further, these diagrams would be used to address the instances in interactions between data entities through the system, helping refine the overall structure.

What is an entity?

An entity could simply be identified as a physical object of the organisation associated with the system. For instance, this could be the employees, customers, scheduled lessons etc. These entities would commonly be represented as tables within a database system.

What is an attribute?

An attribute could be identified as a characteristic of an entity, and would commonly be represented as fields within a database system. Using student entities as an example, the following attributes could include: forename, surname, contact number etc.

What is a primary Key?

A primary key is an attribute used to uniquely identify the various records present within each entity (or instances of the entity). These attributes will automatically provide unique, and incremented values as to individually identify each instance of the entity.

What is a foreign Key?

The term foreign key is used to identify the presence of a Primary key from another entity. These are important attributes, and are used to effectively connect information from two entities, making the database system relational.

Diagram Key



Blue rectangular boxes are used to identify the system entities.



Diamonds represent Actions, and any red identifies their connections with the entities.



These boxes are used to identify the attributes associated with each entity.



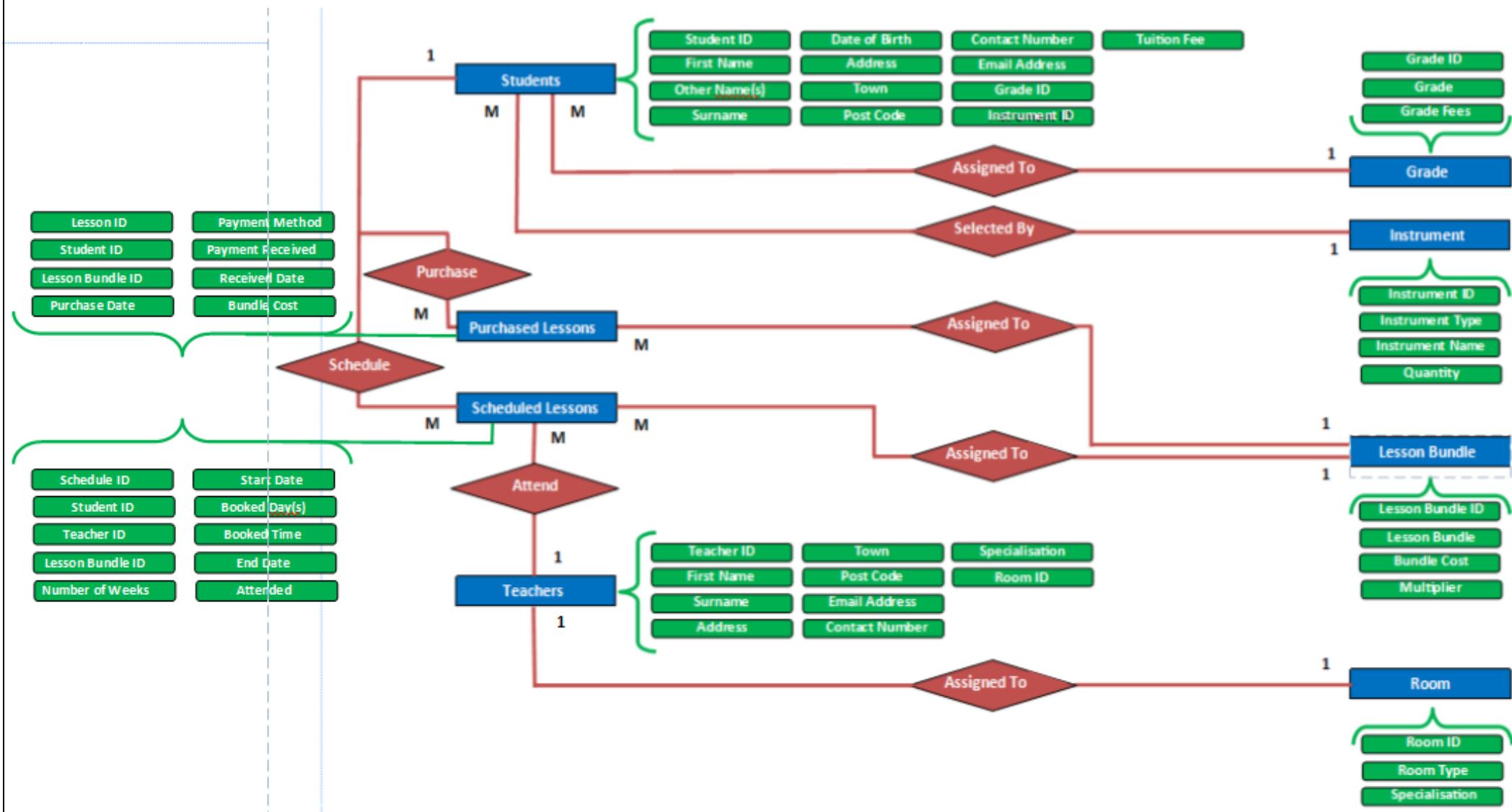
One



Many

Both are used to identify the relationship between the entities

Current System ER Diagram



Explanation of Relationships

There are many relationships which could be established within a relational database system, all of which are based around the following three structures: One-to-one, one-to-many, and many-to many.

- ❖ **One-to-One:** This implies that both tables involved can only possess one record on either side of the relationship, essentially meaning that each value present within the primary key only relates to one record present within the associated table. Using the current system as an example, this relationship has been used within the connection of “Teacher” and “Room” tables, where “Teacher” contains the foreign key. This implies that only a room ID can only be referenced within a “Teacher” record once.

For another comparison, this relationship has been compared to that of a physical partnership. It implies that you can be married, and if you are you will only have one spouse, and so will they.

- ❖ **One-to-Many:** This implies that the foreign key table processes the ability to reference records present within the associated table on numerous occasions. Using the current system as an example, we would be analysing the relationship between “Students”, and “Purchased lessons”, where the latter of which contains the primary key. This implies that there can be numerous lesson records associated with the same student. (The student can purchase many classes).

For an additional comparison, this relationship has been compared to that of a physical relationship between a parent and their children, stating that while the child will only have one mother, the mother can have many children.

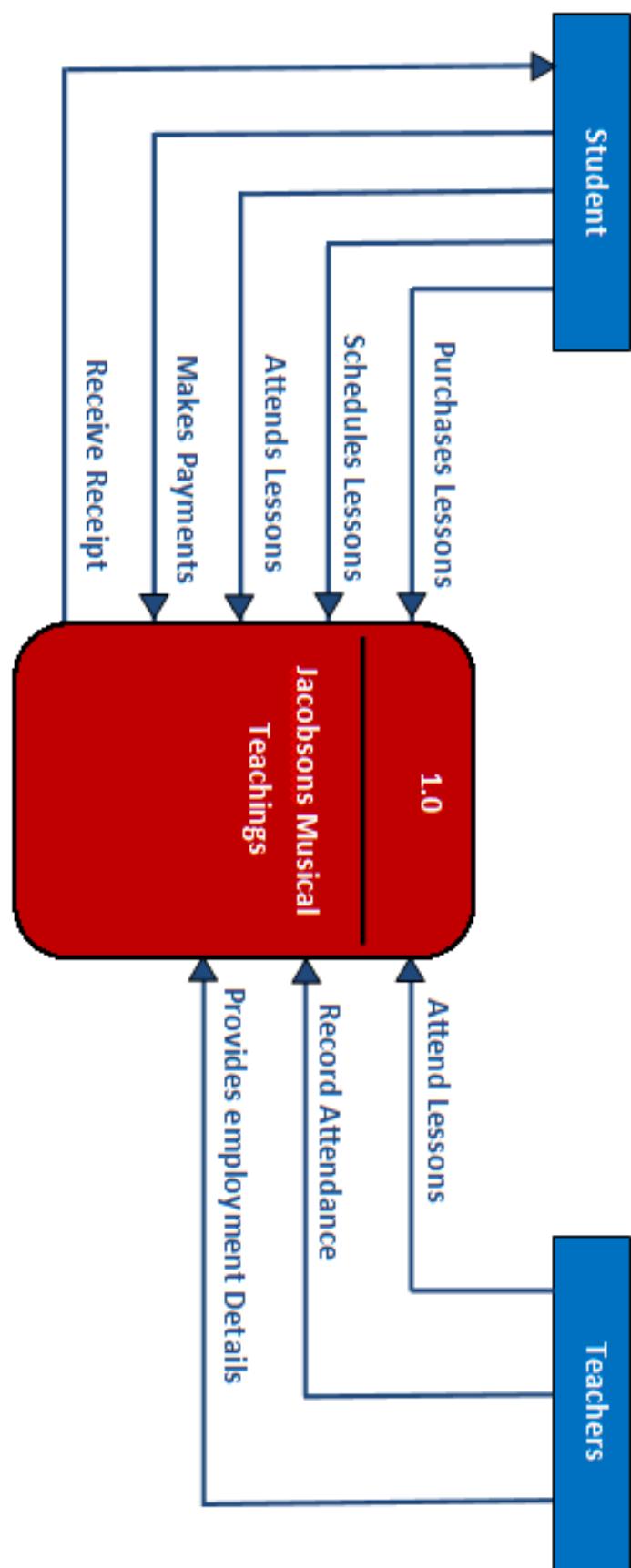
- ❖ **Many-to-Many:** This relationship implies that the records present within both tables can relate to any number of records, in another table. It has been compared to that of siblings, stating that you can have numerous siblings, and that they can also have numerous siblings. It is important to address however, that these relationships cannot be directly accommodated by relational systems, and must instead be linked or associated with a third table, else it may not be operational.

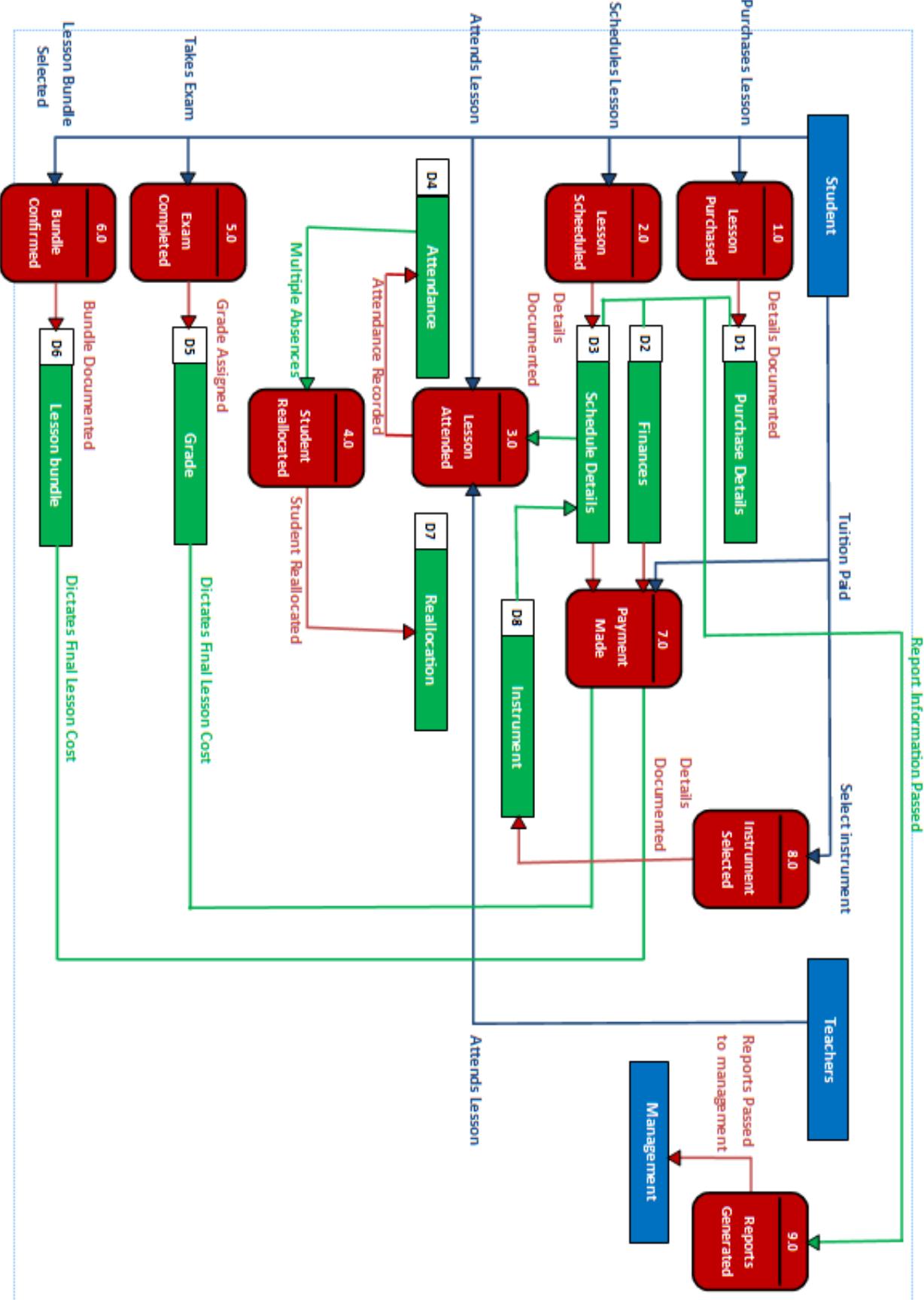
Examples from developed ER Diagram

- ❖ A student can purchase many lessons.
- ❖ A student can schedule many lessons.
- ❖ A grade can be assigned to many students.
- ❖ An instrument can be selected by many students.
- ❖ A lesson bundle can be assigned to many scheduled lessons.
- ❖ A lesson bundle can be assigned to many purchased lessons.
- ❖ A room can be assigned to one teacher.
- ❖ A teacher can attend many lessons.

Appendix 4

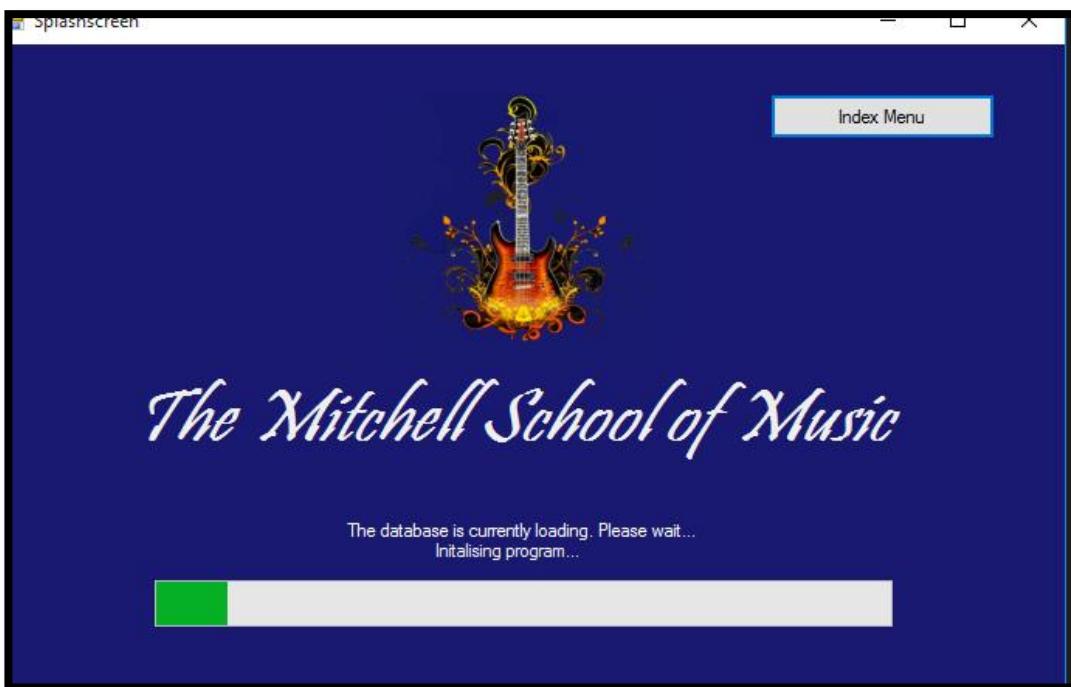
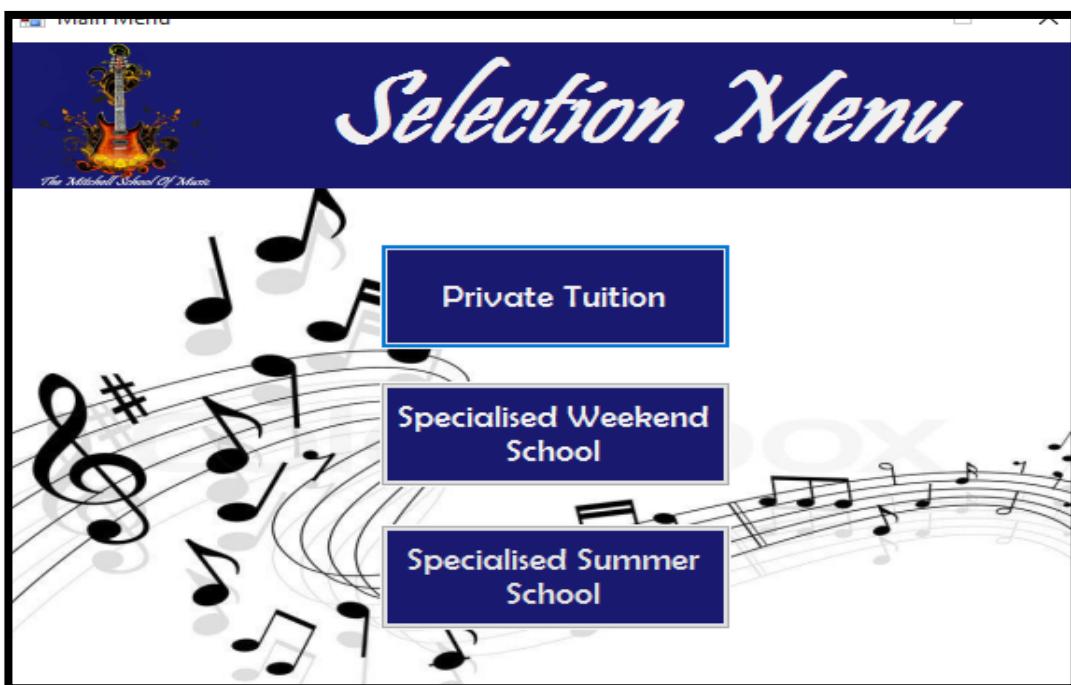
| Data Flow Diagrams |

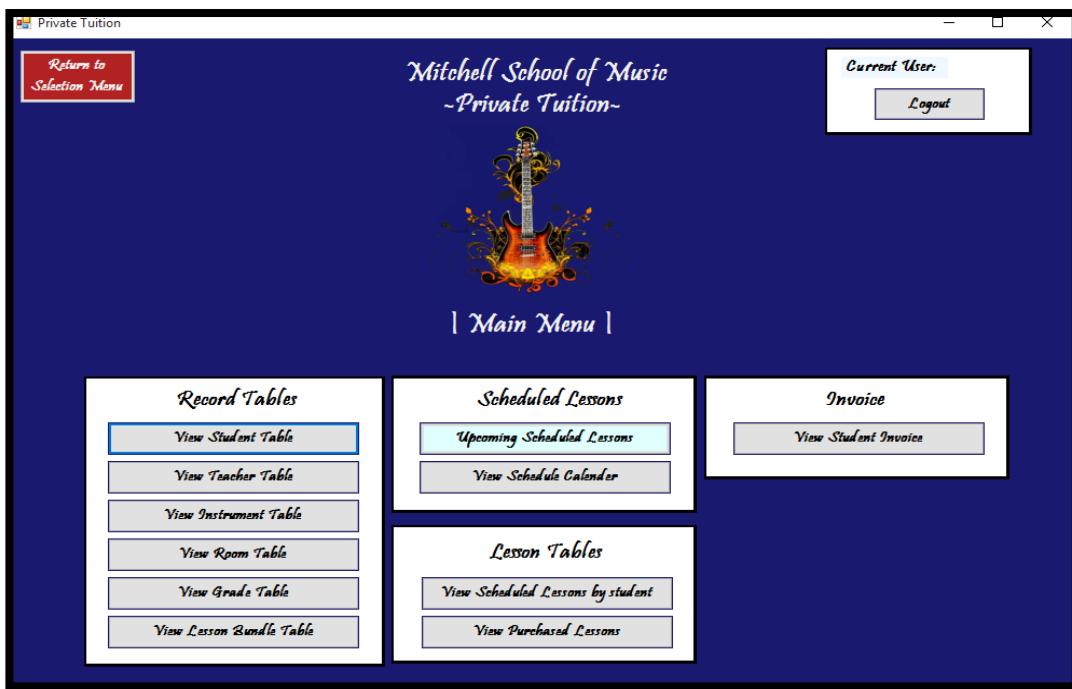
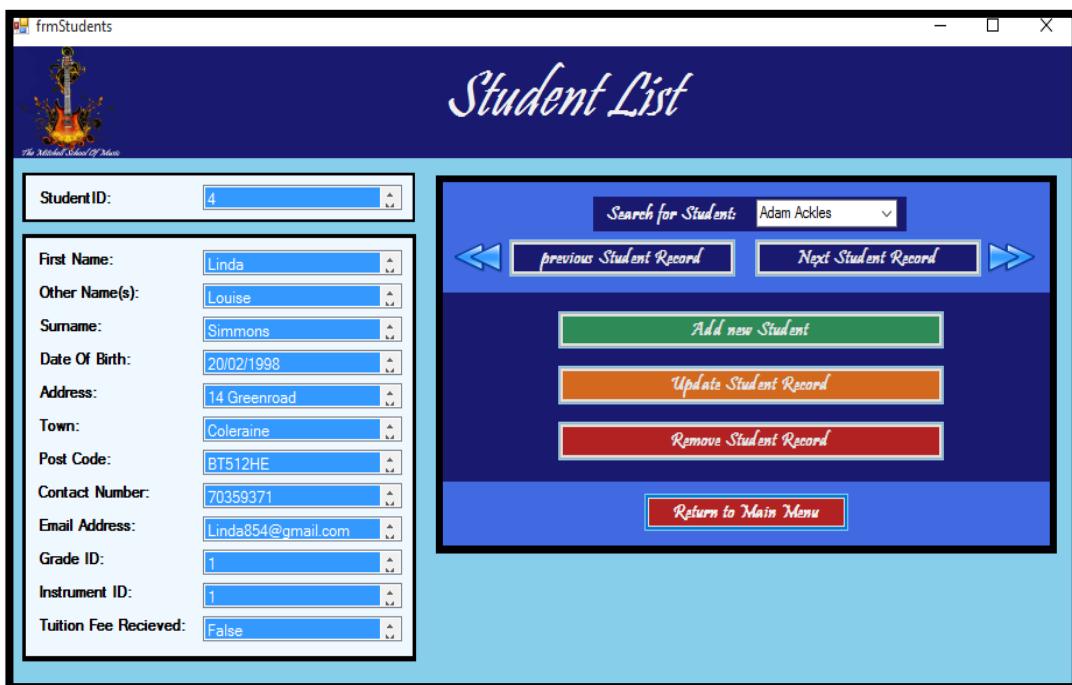
Level 0

Level 1

Appendix 5

| Final System Overview |

Forms**FrmSplash****FrmMain**

FrmPrivateTuitionFrmStudents

FrmTeacher

The FrmTeacher application window features a dark blue header with the title "Teachers List". In the top-left corner is a logo of a guitar with the text "The Musical School Of Music". On the left side, there is a vertical stack of text input fields for teacher details: TeacherID (1), First Name (Sandra), Surname (Smith), Address (362 Atlantic Road), Town (Coleraine), Post Code (BT32 3RE), Contact Number (84933740), Email Address (ssmith362@gmail.co.uk), Specialisation (String), and RoomID (1). To the right of these fields is a central panel with a search bar labeled "Search for Teacher" containing "Sandra", and navigation buttons for "previous Teacher Record" and "Next Teacher Record". Below the search bar are three colored buttons: green for "Add new Teacher", orange for "Update Teacher Record", and red for "Remove Teacher Record". At the bottom of the central panel is a blue button labeled "Return to Main Menu".

FrmInstrument

The FrmInstrument application window features a dark blue header with the title "Instruments List". In the top-left corner is a logo of a guitar with the text "The Musical School Of Music". On the left side, there is a vertical stack of text input fields for instrument details: InstrumentID (1), Instrument Type (String), Instrument Name (Violin), and Quantity (4). To the right of these fields is a central panel with a search bar labeled "Search for Instrument" containing "Violin", and navigation buttons for "previous Instrument Record" and "Next Instrument Record". Below the search bar are three colored buttons: green for "Add new Instrument", orange for "Update Bundle Record", and red for "Remove Instrument Record". At the bottom of the central panel is a blue button labeled "Return to Main Menu".

FrmGrade

frmGrade

Grade List

GradeID: 1

Grade Level: Beginner

Grade Fee:

Search for Grade: Beginner

previous Grade Record Next Grade Record

Add new Grade

Update Grade Record

Remove Grade Record

Return to Main Menu

FrmRoom

frmRoom

Room List

RoomID: 1

Room Type: Practice Room

Specialisation: String

Search for Room: 1

previous Room Record Next Room Record

Add new Room

Update Room Record

Remove Room Record

Return to Main Menu

FrmLessonBundle

frmLessonBundle

Lesson Bundle List



The Mitchell School Of Music

Lesson BundleID:	<input type="text" value="1"/>
Lesson Bundle:	<input type="text" value="10 Lessons"/>
Bundle Cost:	<input type="text" value="10"/>
Multiplier(Discount):	<input type="text" value="1"/>

Search for Bundle: 10 Lessons

previous Lesson Bundle **Next Lesson Bundle**

Add new Bundle

Update Bundle Record

Remove Bundle Record

Return to Main Menu

FrmScheduleTable

frmScheduleTable

Scheduled Lessons List



The Mitchell School Of Music

StudentID:	<input type="text" value="6"/>																		
Student Details <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td>First Name:</td> <td style="text-align: center;"><input type="text" value="Daniel"/></td> </tr> <tr> <td>Surname:</td> <td style="text-align: center;"><input type="text" value="Waters"/></td> </tr> <tr> <td>Contact Number:</td> <td style="text-align: center;"><input type="text" value="37719048"/></td> </tr> </table>		First Name:	<input type="text" value="Daniel"/>	Surname:	<input type="text" value="Waters"/>	Contact Number:	<input type="text" value="37719048"/>												
First Name:	<input type="text" value="Daniel"/>																		
Surname:	<input type="text" value="Waters"/>																		
Contact Number:	<input type="text" value="37719048"/>																		
Teacher Details <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td>TeacherID:</td> <td style="width: 10%; text-align: center;"><input type="text" value="1"/></td> </tr> <tr> <td>First Name:</td> <td style="text-align: center;"><input type="text" value="Sandra"/></td> </tr> <tr> <td>Surname:</td> <td style="text-align: center;"><input type="text" value="Smith"/></td> </tr> <tr> <td>Specialisation:</td> <td style="text-align: center;"><input type="text" value="String"/></td> </tr> <tr> <td>Room:</td> <td style="text-align: center;"><input type="text" value="1"/></td> </tr> </table>		TeacherID:	<input type="text" value="1"/>	First Name:	<input type="text" value="Sandra"/>	Surname:	<input type="text" value="Smith"/>	Specialisation:	<input type="text" value="String"/>	Room:	<input type="text" value="1"/>								
TeacherID:	<input type="text" value="1"/>																		
First Name:	<input type="text" value="Sandra"/>																		
Surname:	<input type="text" value="Smith"/>																		
Specialisation:	<input type="text" value="String"/>																		
Room:	<input type="text" value="1"/>																		
Grade Details <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td>GradeID:</td> <td style="width: 10%; text-align: center;"><input type="text" value="4"/></td> </tr> <tr> <td>Grade:</td> <td style="text-align: center;"><input type="text" value="Diploma"/></td> </tr> </table>		GradeID:	<input type="text" value="4"/>	Grade:	<input type="text" value="Diploma"/>														
GradeID:	<input type="text" value="4"/>																		
Grade:	<input type="text" value="Diploma"/>																		
ScheduleID: <input type="text" value="11"/> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>ScheduleID</th> <th>StudentID</th> <th>TeacherID</th> <th>LessonsPurchasedID</th> <th>Number_Of_Weeks</th> <th>Start_Date</th> <th>Booked_Day(s)</th> <th>Booked_Time</th> <th>End_Date</th> </tr> </thead> <tbody> <tr> <td>11</td> <td>6</td> <td>1</td> <td>3</td> <td>30 Weeks</td> <td>28/10/2016</td> <td>Monday</td> <td>09:30</td> <td>26/05/2017</td> </tr> </tbody> </table>		ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date	11	6	1	3	30 Weeks	28/10/2016	Monday	09:30	26/05/2017
ScheduleID	StudentID	TeacherID	LessonsPurchasedID	Number_Of_Weeks	Start_Date	Booked_Day(s)	Booked_Time	End_Date											
11	6	1	3	30 Weeks	28/10/2016	Monday	09:30	26/05/2017											

Search for Student: Adam Ackles

previous Student Record **Next Student Record**

Add new Student

Update Lesson Record

Remove Lesson Record

Return to Calendar [Placeholder]

Return to Main Menu

PurchasedLessons

PurchasedLessonBundles

Purchased Lessons

StudentID:	4																																									
Student Details																																										
First Name:	Linda																																									
Surname:	Simmons																																									
Contact Number:	70359371																																									
Grade Details																																										
GradeID:	1																																									
Grade:	Beginner																																									
Purchase ID:		13																																								
<table border="1"> <thead> <tr> <th></th> <th>LessonsPurchasedID</th> <th>StudentID</th> <th>LessonBundleID</th> <th>Purchased_Date</th> <th>Payment_Method</th> <th>Payment_Received</th> <th>Payment_Received Date</th> <th>Total_Bundle_Cost</th> <th>Scheduled</th> </tr> </thead> <tbody> <tr> <td>></td> <td>13</td> <td>4</td> <td>2</td> <td>12/08/2016</td> <td>Cash</td> <td><input checked="" type="checkbox"/></td> <td>12/08/2016</td> <td></td> <td><input type="checkbox"/></td> </tr> <tr> <td>*</td> <td>31</td> <td>4</td> <td>3</td> <td>01/08/2016</td> <td>Cash</td> <td><input checked="" type="checkbox"/></td> <td>01/09/2016</td> <td></td> <td><input type="checkbox"/></td> </tr> <tr> <td>*</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td><input type="checkbox"/></td> <td></td> <td></td> <td><input type="checkbox"/></td> </tr> </tbody> </table>				LessonsPurchasedID	StudentID	LessonBundleID	Purchased_Date	Payment_Method	Payment_Received	Payment_Received Date	Total_Bundle_Cost	Scheduled	>	13	4	2	12/08/2016	Cash	<input checked="" type="checkbox"/>	12/08/2016		<input type="checkbox"/>	*	31	4	3	01/08/2016	Cash	<input checked="" type="checkbox"/>	01/09/2016		<input type="checkbox"/>	*						<input type="checkbox"/>			<input type="checkbox"/>
	LessonsPurchasedID	StudentID	LessonBundleID	Purchased_Date	Payment_Method	Payment_Received	Payment_Received Date	Total_Bundle_Cost	Scheduled																																	
>	13	4	2	12/08/2016	Cash	<input checked="" type="checkbox"/>	12/08/2016		<input type="checkbox"/>																																	
*	31	4	3	01/08/2016	Cash	<input checked="" type="checkbox"/>	01/09/2016		<input type="checkbox"/>																																	
*						<input type="checkbox"/>			<input type="checkbox"/>																																	

Search for Student: Adam Ackles

 < previous Student Record | Next Student Record >

 Add New Purchase | Update Student Record | Remove Lesson Record

[Return to Main Menu](#)

FrmAddField*Student*

FrmAddField

Add Student

Add New Student	Return to Students Table
First Name:	<input type="text"/>
Other Name(s):	<input type="text"/>
Surname:	<input type="text"/>
Date Of Birth:	<input type="text"/> 
Address:	<input type="text"/>
Town:	<input type="text"/>
PostCode:	<input type="text"/>
Contact Number:	<input type="text"/>
Email Address:	<input type="text"/>
GradeID	<input type="text"/> 1
InstrumentID	<input type="text"/> 1
Tuition Fee Received	<input type="checkbox"/>

Teacher

frmAddField



Add Teacher

[Add New Teacher](#) [Return to Teacher Table](#)

First Name:

Surname:

Address:

Town:

PostCode:

Email Address:

Contact Number:

Specialisation:

RoomID:

Instrument

frmAddField



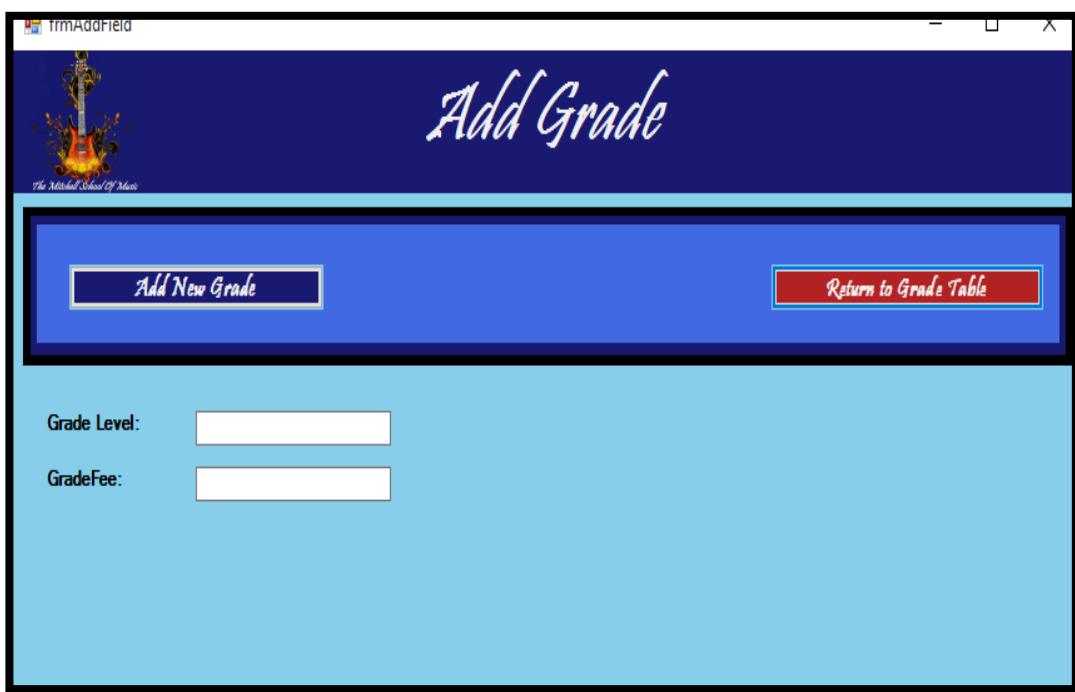
Add Instrument

[Add New Instrument](#) [Return to Instrument Table](#)

Instrument Type:

Instrument Name:

Quantity:

Grade*Lesson Bundle*

Room

frmAddField

Add Room

The Musical School Of Music

[Add New Room](#) [Return to Room Table](#)

Room Type: Class Room

Specialisation: Brass

Scheduled Lessons

frmAddField

Schedule New Lesson

The Musical School Of Music

[Schedule New Lesson](#) [Return to Scheduled Lessons Table](#)

Student ID: 4 | Linda Simmons

Teacher ID: 1 | Sandra Smith

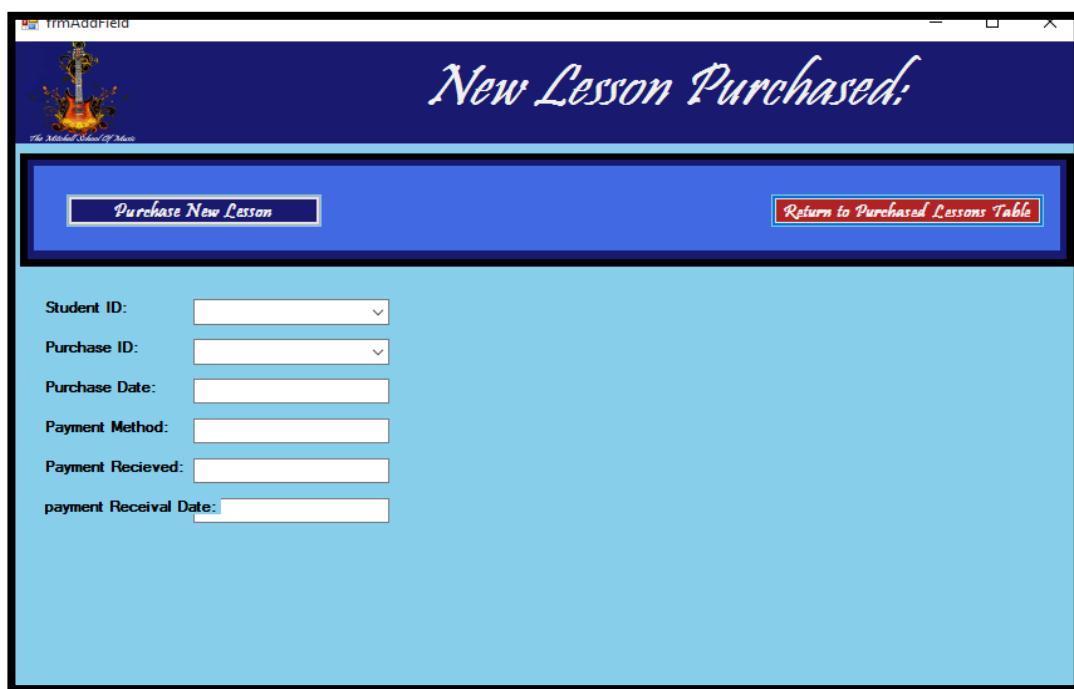
Purchased Lesson:

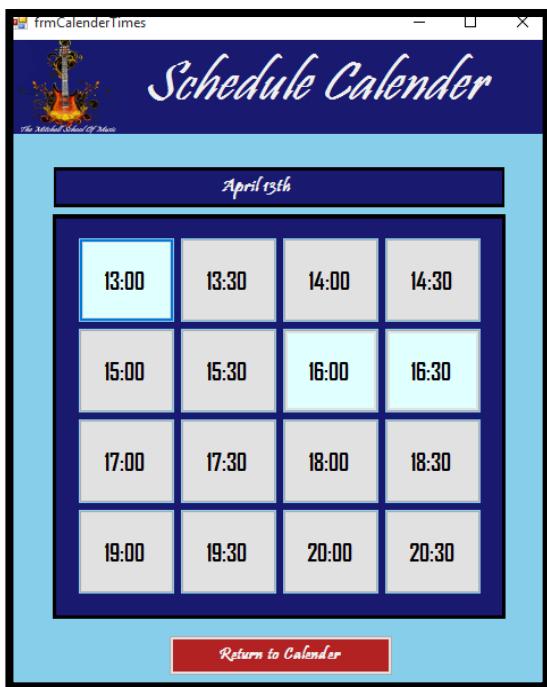
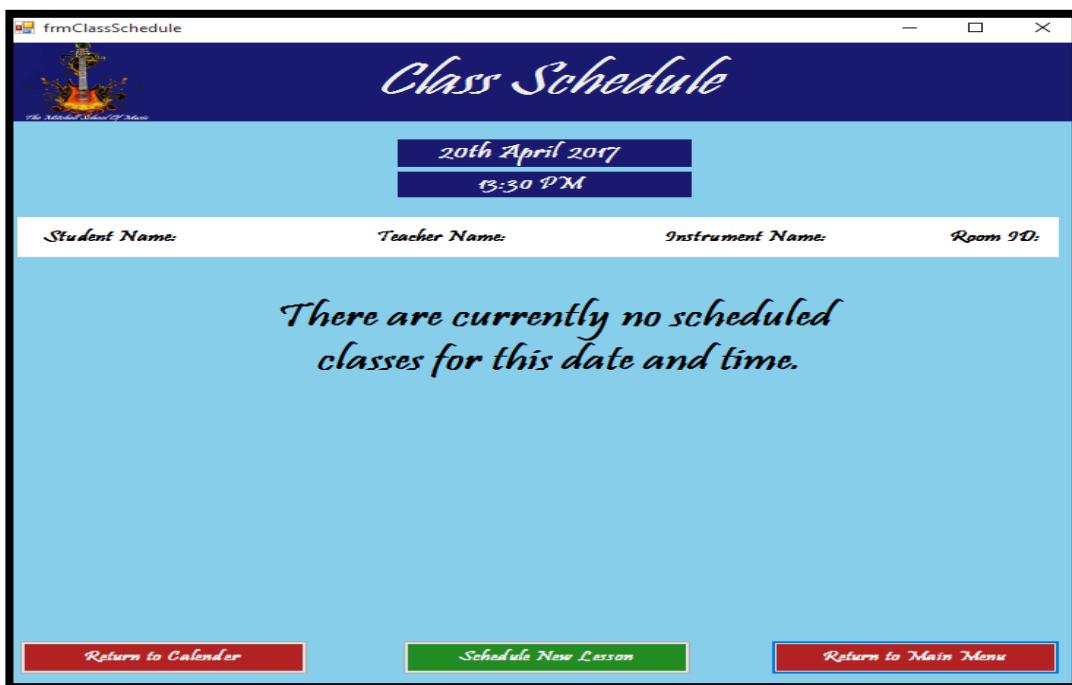
No. of weeks: 5 Weeks

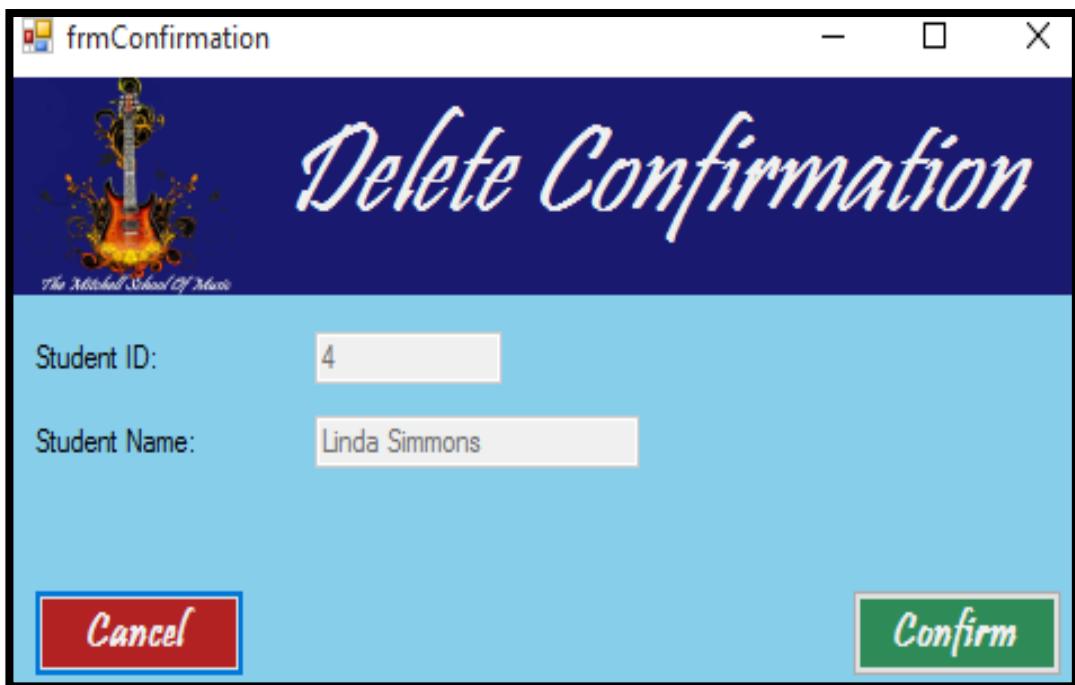
Start Date:

Booked Time: 13:00

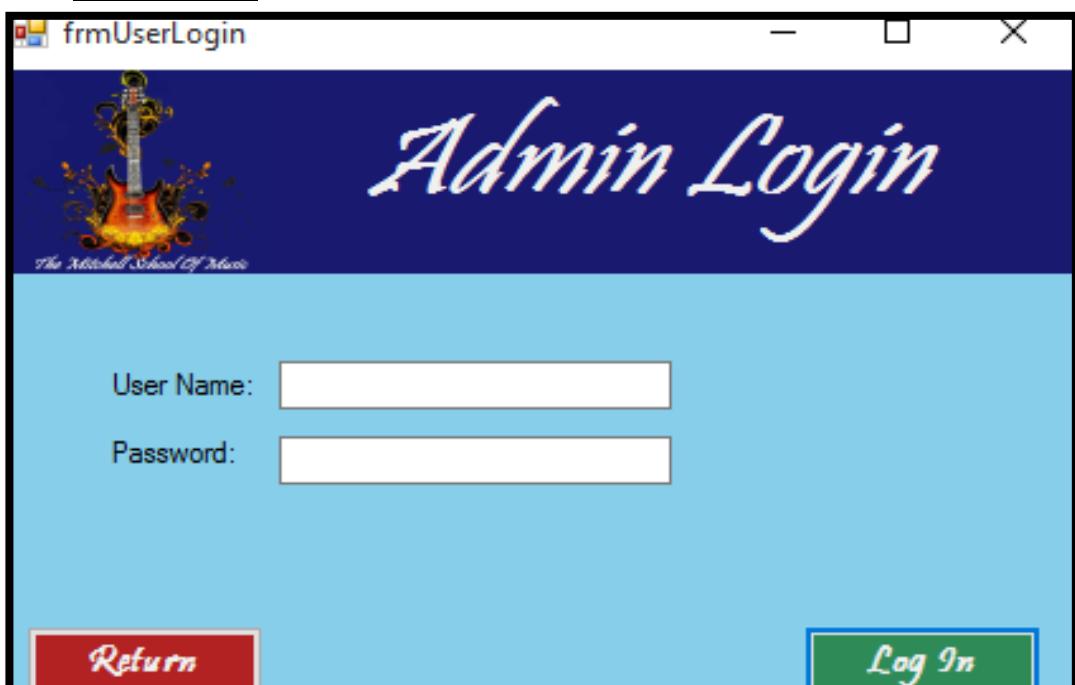
Booked Day(s):

Purchased Lessons*FrmCalendarDates*

FrmCalendar TimesFrmClassSchedule

FrmConfirmation

The frmConfirmation window displays a "Delete Confirmation" message. It features a logo of a guitar on fire in the top left corner. The main title "Delete Confirmation" is centered in a large, stylized font. Below the title, there are two input fields: "Student ID:" with the value "4" and "Student Name:" with the value "Linda Simmons". At the bottom are two buttons: a red "Cancel" button on the left and a green "Confirm" button on the right.

FrmUserLogin

The frmUserLogin window displays an "Admin Login" message. It features a logo of a guitar on fire in the top left corner. The main title "Admin Login" is centered in a large, stylized font. Below the title, there are two input fields: "User Name:" and "Password:", both represented by empty white boxes. At the bottom are two buttons: a red "Return" button on the left and a green "Log In" button on the right.

FrmUpcomingLessons

Upcoming Lessons

Search for: Filter by:

Lesson ID:	Entity Name:	Purchased Lesson ID:	Lesson Date:	Booked Day:	Booked Time:	Attended:
58	Daniel Waters	3	01/05/2017 00:00:00	Monday	09:30	<input type="checkbox"/>
59	Daniel Waters	3	08/05/2017 00:00:00	Monday	09:30	<input type="checkbox"/>
60	Daniel Waters	3	15/05/2017 00:00:00	Monday	09:30	<input type="checkbox"/>
61	Daniel Waters	3	22/05/2017 00:00:00	Monday	09:30	<input type="checkbox"/>

FrmInvoiceStudentSelection

Student Selection

4 | Linda Simmons

FrmInvoiceReport

Purchased Lesson ID:	Lesson Bundle:	Purchased Date:	Bundle Cost:
23	1	01/08/2016	100
33	3	01/08/2016	270

Total Cost: 370

CodeFrmSplash

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;
using System.Configuration;

namespace frmSplash
{
    public partial class frmSplash : Form
    {
        string LoadMessage; // String will contain text.
        string connectionString;
        SqlConnection connection;
        SqlDataReader DataReader;

        List<int> LessonDates_ID = new List<int>();
        List<int> LessonDates_ScheduleID = new List<int>();
        List<DateTime> LessonDates_Dates = new List<DateTime>();
        List<bool> LessonDates_Attended = new List<bool>();
        List<int> LessonDates_ID_Appending = new List<int>();
        List<int> LessonDates_ScheduleID_Appending = new List<int>();
        List<DateTime> LessonDates_Dates_Appending = new List<DateTime>();
        List<bool> LessonDates_Attended_Appending = new List<bool>();
    }
}

```

```
List<int> Students_ID = new List<int>();
List<string> Students_FirstName = new List<string>();
List<string> Students_OtherNames = new List<string>();
List<string> Students_Surname = new List<string>();
List<DateTime> Students_DateOfBirth = new List<DateTime>();
List<string> Students_Address = new List<string>();
List<string> Students_Town = new List<string>();
List<string> Students_PostCode = new List<string>();
List<string> Students_ContactNumber = new List<string>();
List<string> Students_EmailAddress = new List<string>();
List<int> Students_GradeID = new List<int>();
List<int> Students_InstrumentID = new List<int>();
List<bool> Students_Payment = new List<bool>();

List<int> ScheduledLessons_ID = new List<int>();
List<int> ScheduledLessons_StudentID = new List<int>();
List<int> ScheduledLessons_TeacherID = new List<int>();
List<int> ScheduledLessons_PurchasedLessonID = new List<int>();
List<string> ScheduledLessons_NumberOfWeeks = new List<string>();
List<DateTime> ScheduledLessons_StartDate = new List<DateTime>();
List<string> ScheduledLessons_BookedDays = new List<string>();
List<string> ScheduledLessons_BookedTime = new List<string>();
List<DateTime> ScheduledLessons_EndDate = new List<DateTime>();

bool ScheduledLessons_AppendComplete;

List<string> Reallocation_Students_FirstName = new List<string>();
List<string> Reallocation_Students_OtherNames = new List<string>();
List<string> Reallocation_Students_Surname = new List<string>();
List<DateTime> Reallocation_Students_DateOfBirth = new List<DateTime>();
List<string> Reallocation_Students_Address = new List<string>();
List<string> Reallocation_Students_Town = new List<string>();
List<string> Reallocation_Students_PostCode = new List<string>();
List<string> Reallocation_Students_ContactNumber = new List<string>();
List<string> Reallocation_Students_EmailAddress = new List<string>();
List<int> Reallocation_Students_GradeID = new List<int>();
List<int> Reallocation_Students_InstrumentID = new List<int>();
List<bool> Reallocation_Students_Payment = new List<bool>();
List<int> Rellocation_ScheduleLessonsID = new List<int>();
List<int> Reallocation_StudentAbsences = new List<int>();
List<int> Reallocation_LessonDates_ScheduleID = new List<int>();
List<bool> Reallocation_LessonDates_Attended = new List<bool>();
List<DateTime> Reallocation_LessonDates_Dates = new List<DateTime>();
List<int> Reallocation_Desired_ScheduleIDs = new List<int>();
List<int> Reallocation_Desired_StudentIDs = new List<int>();

List<int> LessonDate_Archive_ID = new List<int>();
List<int> LessonDate_Archive_ScheduleID = new List<int>();
List<DateTime> LessonDate_Archive_Date = new List<DateTime>();
List<bool> LessonDate_Archive_Attended = new List<bool>();

DateTime CurrentDate;
DateTime Reference;
bool Completed;

public frmSplash()
{
    InitializeComponent();
```

```
        connectionString =
ConfigurationManager.ConnectionStrings["frmSplash.Properties.Settings.PrivateTuitionCo
nnectionString"].ConnectionString;

        // This initialises the clock present within the form.
tLoad.Start();

Establish_DataBase_Connection();
LocalStorage_LessonDates();
LocalStorage_LessonDate_Archive();
LocalStorage_ScheduledLessons();
LocalStorage_Students();
Schedule_Archive();
Student_Reallocation();
}

// LOAD TIMER
private void tLoad_Tick(object sender, EventArgs e)
{
    LoadingBar.Increment(2); // This will cause the loading bar to increase
each time the clock ticks.
    UpdateLoadMessage(); // This line of code will run the updateLoadMessage
function, each tick.
}

// LOADING PROGRESS OUTCOMES
public void UpdateLoadMessage()
{
    // This function will be used to identify the progress of the loading bar,
    // and will update the loading message accordingly.

    if (LoadingBar.Value > 0 && LoadingBar.Value < 25)
    {
        LoadMessage = "Initialising program...";

    }
    else if (LoadingBar.Value > 25 && LoadingBar.Value < 75)
    {
        LoadMessage = "Organising files...";
        Archive_Passed_LessonDates();
    }
    else if (LoadingBar.Value > 75)
    {
        LoadMessage = "Finalising program...";

    }

    if (LoadingBar.Value == 100)
    {
        OpenMenu(); // This line of code is used to run the OpenMenu function.
    }

    lblLoadingProgress.Text = LoadMessage; // Update the label accordingly.
}

// ESTABLISH DATABASE CONNECTION
public void Establish_DataBase_Connection()
{
    try
    {
        using (connection = new SqlConnection(connectionString))
        {
            connection.Open();
```

```
        }
    }
    catch (Exception ex)
    {
        try
        {
            using (connection = new SqlConnection(connectionString))
            {
                connection.Open();
            }
        }
        catch (Exception ex2)
        {
            MessageBox.Show("Connection Error.");
        }
    }
}

// OPEN MAIN MENU
public void OpenMenu()
{
    // This function will attempt to open the Main menu.
    if (LoadingBar.Value == 100) // Checks the progress bar.
    {
        tLoad.Stop(); // Disables timer.
        frmMain MainMenu = new frmMain(); // Assigns local value to main form.
        MainMenu.Show(); // Opens Main form.
        this.Hide(); // Closes this form.

    }
}

// TEST BUTTON
private void btnIndex_Click(object sender, EventArgs e)
{
    // This is merely a testing program.
    GlobalVariables.UserLoggedIn = true;

    frmPrivateTuition Mainmenu = new frmPrivateTuition();
    Mainmenu.Show();
    this.Hide();
    tLoad.Stop();
}

// CLOSE SYSTEM
private void frmSplash_FormClosing(object sender, FormClosingEventArgs e)
{
    Application.Exit();
    this.Close();
}

// STORE LESSON DATE INFORMATION
public void LocalStorage_LessonDates()
{
    using (connection = new SqlConnection(connectionString))
    {
        connection.Open();
        using (SqlCommand cmd = new SqlCommand("SELECT * FROM LessonDates",
connection))
        {

```

```
DataReader = cmd.ExecuteReader();
while (DataReader.Read())
{
    LessonDates_ID.Add(DataReader.GetInt32(0));
    LessonDates_ScheduleID.Add(DataReader.GetInt32(1));
    LessonDates_Dates.Add(DataReader.GetDateTime(2));
    LessonDates_Attended.Add(DataReader.GetBoolean(3));

    Reallocation_LessonDates_ScheduleID.Add(DataReader.GetInt32(1));
    Reallocation_LessonDates_Dates.Add(DataReader.GetDateTime(2));

    Reallocation_LessonDates_Attended.Add(DataReader.GetBoolean(3));
}
}

// STORE STUDENT INFORMATION
public void LocalStorage_Students()
{
    using (connection = new SqlConnection(connectionString))
    {
        connection.Open();
        using (SqlCommand cmd = new SqlCommand("SELECT * FROM Students",
connection))
        {

            DataReader = cmd.ExecuteReader();
            while (DataReader.Read())
            {
                Students_ID.Add(DataReader.GetInt32(0));
                Students_FirstName.Add(DataReader.GetString(1));
                Students_OtherNames.Add(DataReader.GetString(2));
                Students_Surname.Add(DataReader.GetString(3));
                Students_DateOfBirth.Add(DataReader.GetDateTime(4));
                Students_Address.Add(DataReader.GetString(5));
                Students_Town.Add(DataReader.GetString(6));
                Students_PostCode.Add(DataReader.GetString(7));
                Students_ContactNumber.Add(DataReader.GetString(8));
                Students_EmailAddress.Add(DataReader.GetString(9));
                Students_GradeID.Add(DataReader.GetInt32(10));
                Students_InstrumentID.Add(DataReader.GetInt32(11));
                Students_Payment.Add(DataReader.GetBoolean(12));
                Reallocation_StudentAbsences.Add(0);
            }
        }
    }
}

// STORE SCHEDULED LESSONS INFORMATION
public void LocalStorage_ScheduledLessons()
{
    using (connection = new SqlConnection(connectionString))
    {
        connection.Open();
        using (SqlCommand cmd = new SqlCommand("SELECT * FROM
Scheduled_Lessons", connection))
        {

            DataReader = cmd.ExecuteReader();
            while (DataReader.Read())
```

```
{  
    ScheduledLessons_ID.Add(DataReader.GetInt32(0));  
    ScheduledLessons_StudentID.Add(DataReader.GetInt32(1));  
    ScheduledLessons_TeacherID.Add(DataReader.GetInt32(2));  
  
    ScheduledLessons_PurchasedLessonID.Add(DataReader.GetInt32(3));  
    ScheduledLessons_NumberOfWeeks.Add(DataReader.GetString(4));  
    ScheduledLessons_StartDate.Add(DataReader.GetDateTime(5));  
    ScheduledLessons_BookedDays.Add(DataReader.GetString(6));  
    ScheduledLessons_BookedTime.Add(DataReader.GetString(7));  
    ScheduledLessons_EndDate.Add(DataReader.GetDateTime(8));  
}  
}  
}  
}  
  
// STORE LESSON DATE ARCHIVE INFORMATION  
public void LocalStorage_LessonDate_Archive()  
{  
    using (connection = new SqlConnection(connectionString))  
    {  
        connection.Open();  
        using (SqlCommand cmd = new SqlCommand("SELECT * FROM  
LessonDate_Archive", connection))  
        {  
  
            DataReader = cmd.ExecuteReader();  
            while (DataReader.Read())  
            {  
                LessonDate_Archive_ID.Add(DataReader.GetInt32(0));  
                LessonDate_Archive_ScheduleID.Add(DataReader.GetInt32(2));  
                LessonDate_Archive_Date.Add(DataReader.GetDateTime(3));  
                LessonDate_Archive_Attended.Add(DataReader.GetBoolean(4));  
  
                Reallocation_LessonDates_ScheduleID.Add(DataReader.GetInt32(2));  
                Reallocation_LessonDates_Dates.Add(DataReader.GetDateTime(3));  
  
                Reallocation_LessonDates_Attended.Add(DataReader.GetBoolean(4));  
            }  
        }  
    }  
}  
  
// ARCHIVE COMPLETE LESSONS  
public void Archive_Passed_LessonDates()  
{  
    if (Completed == false)  
    {  
        Completed = true;  
  
        for (int x = 0; x < LessonDates_ID.Count(); x++)  
        {  
            CurrentDate = DateTime.Today;  
            if (LessonDates_Dates[x] < CurrentDate)  
            {  
                LessonDates_ID_Appending.Add(LessonDates_ID[x]);  
            }  
        }  
  
        try  
        {  
    }
```

```
        for (int x = 0; x < LessonDates_ID_Appending.Count(); x++)
    {
        for (int y = 0; y < LessonDates_ID.Count(); y++)
        {
            if (LessonDates_ID[y] == LessonDates_ID_Appending[x])
            {

LessonDates_ScheduleID_Appending.Add(LessonDates_ScheduleID[y]);
                LessonDates_Dates_Appending.Add(LessonDates_Dates[y]);

LessonDates_Attended_Appending.Add(LessonDates_Attended[y]);
            }
        }
    }

        for (int x = 0; x < LessonDates_ID_Appending.Count(); x++)
    {
        using (connection = new SqlConnection(connectionString))
        {
            connection.Open();
            using (SqlCommand cmd = new SqlCommand("INSERT INTO
LessonDate_Archive (LessonDatesID, ScheduleID, Date, Attended) VALUES (@LessonDateID,
@ScheduleID, @Date, @Attended)", connection))
            {
                cmd.Parameters.AddWithValue("@LessonDateID",
LessonDates_ID_Appending[x]);
                cmd.Parameters.AddWithValue("@ScheduleID",
LessonDates_ScheduleID[x]);
                cmd.Parameters.AddWithValue("@Date",
LessonDates_Dates_Appending[x]);
                cmd.Parameters.AddWithValue("@Attended",
LessonDates_Attended_Appending[x]);

                cmd.ExecuteNonQuery();
            }

            using (SqlCommand cnd = new SqlCommand("DELETE FROM
LessonDates WHERE LessonDatesID = @LessonDateID", connection))
            {
                cnd.Parameters.AddWithValue("@LessonDateID",
LessonDates_ID_Appending[x]);

                cnd.ExecuteNonQuery();
            }
        }
    }
}
catch (Exception ex)
{
    MessageBox.Show("Error");
}

Archieve_Clear_OldRecords();
Archieve_Clear_OldRecords();
}

// CLEAR OLD ARCHIVED LESSON DATES
public void Archieve_Clear_OldRecords()
```

```
{  
    CurrentDate = DateTime.Today;  
    CurrentDate = CurrentDate.AddMonths(-2);  
  
    for (int x = 0; x < LessonDate_Archive_ID.Count(); x++)  
    {  
        if (LessonDate_Archive_Date[x] < CurrentDate)  
        {  
            using (connection = new SqlConnection(connectionString))  
            {  
                using (SqlCommand cmd = new SqlCommand("DELETE FROM  
LessonDate_Archive WHERE LessonDates_ArchiveID = @LessonDateID", connection))  
                {  
                    connection.Open();  
                    cmd.Parameters.AddWithValue("@LessonDateID",  
LessonDate_Archive_ID[x]);  
  
                    cmd.ExecuteNonQuery();  
                }  
            }  
        }  
    }  
  
    // REALLOCATE STUDENTS  
    public void Student_Reallocation()  
    {  
        CurrentDate = DateTime.Today;  
        Reference = CurrentDate;  
        CurrentDate = CurrentDate.AddMonths(-1);  
  
        // Identify lesson values which have been completed within the last month.  
        for (int x = 0; x < Reallocation_LessonDates_ScheduleID.Count(); x++)  
        {  
            // Any lessons that fall outside of the previous month, will be  
            removed from storage.  
            // This essentially means that they will not be checked.  
            if (Reallocation_LessonDates_Dates[x] < CurrentDate ||  
Reallocation_LessonDates_Dates[x] > Reference)  
            {  
                Reallocation_LessonDates_ScheduleID.RemoveAt(x);  
                Reallocation_LessonDates_Dates.RemoveAt(x);  
                Reallocation_LessonDates_Attended.RemoveAt(x);  
                x -= 1;  
            }  
        }  
  
        for (int x = 0; x < Reallocation_LessonDates_ScheduleID.Count(); x++)  
        {  
            for (int y = 0; y < ScheduledLessons_ID.Count(); y++)  
            {  
                if (Reallocation_LessonDates_ScheduleID[x] ==  
ScheduledLessons_ID[y])  
                {  
  
                    Reallocation_Desired_ScheduleIDs.Add(Reallocation_LessonDates_ScheduleID[x]);  
  
                }  
            }  
        }  
    }  
}
```

```
for (int x = 0; x < ScheduledLessons_ID.Count(); x++)
{
    for (int y = 0; y < Reallocation_Desired_ScheduleIDs.Count(); y++)
    {
        if (ScheduledLessons_ID[x] == Reallocation_Desired_ScheduleIDs[y])
        {

Reallocation_Desired_StudentIDs.Add(ScheduledLessons_StudentID[x]);
        }
    }
}

for (int x = 0; x < Reallocation_Desired_StudentIDs.Count(); x++)
{
    for (int y = 0; y < Students_ID.Count(); y++)
    {
        if (Reallocation_Desired_StudentIDs[x] == Students_ID[y])
        {
            Reallocation_StudentAbsences[y] += 1;
        }
    }
}

int r = 0;
for (int x = 0; x < Reallocation_Desired_StudentIDs.Count(); x++)
{
    while (r < Students_ID.Count())
    {
        if (Reallocation_StudentAbsences[r] > 3)
        {
            Reallocation_Students_FirstName.Add(Students_FirstName[r]);
            Reallocation_Students_OtherNames.Add(Students_OtherNames[r]);
            Reallocation_Students_Surname.Add(Students_Surname[r]);

Reallocation_Students_DateOfBirth.Add(Students_DateOfBirth[r]);
            Reallocation_Students_Address.Add(Students_Address[r]);
            Reallocation_Students_Town.Add(Students_Town[r]);
            Reallocation_Students_PostCode.Add(Students_PostCode[r]);

Reallocation_Students_ContactNumber.Add(Students_ContactNumber[r]);

Reallocation_Students_EmailAddress.Add(Students_EmailAddress[r]);
            Reallocation_Students_GradeID.Add(Students_GradeID[r]);

Reallocation_Students_InstrumentID.Add(Students_InstrumentID[r]);
            Reallocation_Students_Payment.Add(Students_Payment[r]);
            r++;
        }
        else
        {
            r++;
        }
    }
}

//  

//  

//  

//  

//
```

```
}

// SCHEDULE ARCHIVE
public void Schedule_Archive()
{
    CurrentDate = DateTime.Today;
    for (int x = 0; x < ScheduledLessons_ID.Count(); x++)
    {
        ScheduledLessons_AppendComplete = false;

        if (ScheduledLessons_EndDate[x] < CurrentDate)
        {
            if (ScheduledLessons_EndDate != null && ScheduledLessons_StartDate
!= null)
            {

                using (connection = new SqlConnection(connectionString))
                {
                    connection.Open();
                    using (SqlCommand cmd = new SqlCommand("INSERT INTO
Archive_ScheduledLessons (ScheduleID, StudentID, TeacherID, Purchased_LessonID,
Number_Of_Weeks, Start_Date, Booked_Days, Booked_Time, End_Date) VALUES (@ScheduleID,
@StudentID, @TeacherID, @Purchased_LessonsID, @Number_Of_Weeks, @Start_Date,
@Booked_Days, @Booked_Time, @End_Date)", connection))
                    {
                        cmd.Parameters.AddWithValue("@ScheduleID",
ScheduledLessons_ID[x]);
                        cmd.Parameters.AddWithValue("@StudentID",
ScheduledLessons_StudentID[x]);
                        cmd.Parameters.AddWithValue("@TeacherID",
ScheduledLessons_TeacherID[x]);
                        cmd.Parameters.AddWithValue("@Purchased_LessonsID",
ScheduledLessons_PurchasedLessonID[x]);
                        cmd.Parameters.AddWithValue("@Number_Of_Weeks",
ScheduledLessons_NumberOfWeeks[x]);
                        cmd.Parameters.AddWithValue("@Start_Date",
ScheduledLessons_StartDate[x]);
                        cmd.Parameters.AddWithValue("@Booked_Days",
ScheduledLessons_BookedDays[x]);
                        cmd.Parameters.AddWithValue("@Booked_Time",
ScheduledLessons_BookedTime[x]);
                        cmd.Parameters.AddWithValue("@End_Date",
ScheduledLessons_EndDate[x]);

                        cmd.ExecuteNonQuery();
                        ScheduledLessons_AppendComplete = true;
                    }

                    if (ScheduledLessons_AppendComplete == true)
                    {
                        using (SqlCommand cnd = new SqlCommand("DELETE FROM
Scheduled_Lessons WHERE ScheduleID = @ScheduleID", connection))
                        {
                            cnd.Parameters.AddWithValue("@ScheduleID",
ScheduledLessons_ID[x]);
                            cnd.ExecuteNonQuery();
                        }
                    }
                }
            }
        }
    }
}
```

```
        {
            using (SqlCommand cmd = new SqlCommand("DELETE FROM
Archive_ScheduledLessons WHERE ScheduleID = @ScheduleID", connection))
            {
                cmd.Parameters.AddWithValue("@ScheduleID",
ScheduledLessons_ID[x]);
                cmd.ExecuteNonQuery();
                MessageBox.Show("Archive Failed");
            }
        }
    }
}
```

FrmMain

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace frmSplash
{
    public partial class frmMain : Form
    {
        public frmMain()
        {
            InitializeComponent();
        }

        private void btnPrivateTuition_Click(object sender, EventArgs e)
        {
            frmPrivateTuition PrivateT = new frmPrivateTuition();
            PrivateT.Show();
            this.Close();
        }

        private void btnWeekendSchool_Click(object sender, EventArgs e)
        {
            MessageBox.Show("System Is In Development. Please View 'Private Tuition'.");
        }

        private void btnSummerSchool_Click(object sender, EventArgs e)
        {
            MessageBox.Show("System Is In Development. Please View 'Private Tuition'.");
        }

        // BUTTON FEEDBACK
        // Private Tuition
        private void btnPrivateTuition_MouseEnter(object sender, EventArgs e)
        {
            btnPrivateTuition.BackColor = Color.LightSlateGray;
            this.Cursor = Cursors.Hand;
        }
        private void btnPrivateTuition_MouseLeave(object sender, EventArgs e)
        {
            btnPrivateTuition.BackColor = Color.MidnightBlue;
            this.Cursor = Cursors.Arrow;
        }

        // Weekend School
        private void btnWeekendSchool_MouseEnter(object sender, EventArgs e)
        {
            btnWeekendSchool.BackColor = Color.LightSlateGray;
            this.Cursor = Cursors.Hand;
        }
        private void btnWeekendSchool_MouseLeave(object sender, EventArgs e)
        {
```

```
        btnWeekendSchool.BackColor = Color.MidnightBlue;
        this.Cursor = Cursors.Arrow;
    }

    // Summer School
    private void btnSummerSchool_MouseEnter(object sender, EventArgs e)
    {
        btnSummerSchool.BackColor = Color.LightSlateGray;
        this.Cursor = Cursors.Hand;
    }
    private void btnSummerSchool_MouseLeave(object sender, EventArgs e)
    {
        btnSummerSchool.BackColor = Color.MidnightBlue;
        this.Cursor = Cursors.Arrow;
    }
}
```

FrmPrivateTuition

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Configuration;
using System.Data.SqlClient;

namespace frmSplash
{
    public partial class frmPrivateTuition : Form
    {
        public frmPrivateTuition()
        {
            InitializeComponent();
            if (GlobalVariables.UserLoggedIn == true)
            {
                lblUserLoginDetail.Text = "Current User: " + GlobalVariables.Username;
                btnLogin.Text = "Logout";
            }
            else
            {
                lblUserLoginDetail.Text = "No Current Admin";
            }
        }

        private void btnReturn_Click(object sender, EventArgs e)
        {
            frmMain MainForm = new frmMain(); // Assigning the form to a referencable
            value within this class.
            MainForm.Show(); // Then calling this value, showing the form.
            this.Hide(); // Then hiding this form.
        }

        private void btnStudents_Click(object sender, EventArgs e)
        {
            GlobalVariables.StudentForm.Show();
            this.Hide();
        }

        private void btnTeachers_Click(object sender, EventArgs e)
        {
            GlobalVariables.TeacherForm.Show();
            this.Hide();
        }

        private void btnInstrument_Click(object sender, EventArgs e)
        {
            GlobalVariables.InstrumentForm.Show();
            this.Hide();
        }

        private void btnRoom_Click(object sender, EventArgs e)
        {
            GlobalVariables.RoomForm.Show();
            this.Hide();
        }
```

```
}

private void btnGrade_Click(object sender, EventArgs e)
{
    GlobalVariables.GradeForm.Show();
    this.Hide();
}

private void btnLessonBundle_Click(object sender, EventArgs e)
{
    GlobalVariables.LessonBundleForm.Show();
    this.Hide();
}

private void btnCalender_Click(object sender, EventArgs e)
{
    GlobalVariables.Calendar.Show();
    this.Hide();
}

private void btnScheduledLessons_Click(object sender, EventArgs e)
{
    GlobalVariables.ScheduledLessonForm.Show();
    this.Hide();
}

private void btnPurchasedLessons_Click(object sender, EventArgs e)
{
    GlobalVariables.PurchasedLessonForm.Show();
    this.Hide();
}

private void frmPrivateTuition_FormClosing(object sender, FormClosingEventArgs e)
{
    Application.ExitThread();
}

private void btnLogin_Click(object sender, EventArgs e)
{
    GlobalVariables.PreviousForm = "Menu";

    if (GlobalVariables.UserLoggedIn == true)
    {
        GlobalVariables.Username = "";
        GlobalVariables.UserLoggedIn = false;
        btnLogin.Text = "Login";
        lblUserLoginDetail.Text = "No Current Admin";
    }
    else
    {
        frmUserLogin Login = new frmUserLogin();
        Login.Show();
        this.Hide();
    }
}

private void button1_Click(object sender, EventArgs e)
{
```

```
}

private void button1_Click_1(object sender, EventArgs e)
{
    frmViewUpcomingLessons upcomingLessons = new frmViewUpcomingLessons();
    upcomingLessons.Show();
    this.Hide();
}

private void button2_Click(object sender, EventArgs e)
{
    frmArchiveSchedule Archive = new frmArchiveSchedule();
    Archive.Show();
    this.Hide();
}

// BUTTON FEEDBACK
// Student Button
private void btnStudents_MouseEnter(object sender, EventArgs e)
{
    btnStudents.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}
private void btnStudents_MouseLeave(object sender, EventArgs e)
{
    btnStudents.BackColor = Color.LightCyan;
    this.Cursor = Cursors.Arrow;
}

// Teacher Button
private void btnTeachers_MouseEnter(object sender, EventArgs e)
{
    btnTeachers.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}
private void btnTeachers_MouseLeave(object sender, EventArgs e)
{
    btnTeachers.BackColor = Color.LightCyan;
    this.Cursor = Cursors.Arrow;
}

// Instrument Button
private void btnInstrument_MouseEnter(object sender, EventArgs e)
{
    btnInstrument.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}
private void btnInstrument_MouseLeave(object sender, EventArgs e)
{
    btnInstrument.BackColor = Color.LightCyan;
    this.Cursor = Cursors.Arrow;
}

// Room Button
private void btnRoom_MouseEnter(object sender, EventArgs e)
{
    btnRoom.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}
private void btnRoom_MouseLeave(object sender, EventArgs e)
{
    btnRoom.BackColor = Color.LightCyan;
```

```
        this.Cursor = Cursors.Arrow;
    }

    // Grade Button
    private void btnGrade_MouseEnter(object sender, EventArgs e)
    {
        btnGrade.BackColor = Color.LightSlateGray;
        this.Cursor = Cursors.Hand;
    }
    private void btnGrade_MouseLeave(object sender, EventArgs e)
    {
        btnGrade.BackColor = Color.LightCyan;
        this.Cursor = Cursors.Arrow;
    }

    // LessonBundle Button
    private void btnLessonBundle_MouseEnter(object sender, EventArgs e)
    {
        btnLessonBundle.BackColor = Color.LightSlateGray;
        this.Cursor = Cursors.Hand;
    }
    private void btnLessonBundle_MouseLeave(object sender, EventArgs e)
    {
        btnLessonBundle.BackColor = Color.LightCyan;
        this.Cursor = Cursors.Arrow;
    }

    // Upcoming Lesson Bundles Button
    private void button1_MouseEnter(object sender, EventArgs e)
    {
        button1.BackColor = Color.LightSlateGray;
        this.Cursor = Cursors.Hand;
    }
    private void button1_MouseLeave(object sender, EventArgs e)
    {
        button1.BackColor = Color.LightCyan;
        this.Cursor = Cursors.Arrow;
    }

    // Calendar Button
    private void btnCalender_MouseEnter(object sender, EventArgs e)
    {
        btnCalender.BackColor = Color.LightSlateGray;
        this.Cursor = Cursors.Hand;
    }
    private void btnCalender_MouseLeave(object sender, EventArgs e)
    {
        btnCalender.BackColor = Color.LightCyan;
        this.Cursor = Cursors.Arrow;
    }

    // Scheduled Lessons Button
    private void btnScheduledLessons_MouseEnter(object sender, EventArgs e)
    {
        btnScheduledLessons.BackColor = Color.LightSlateGray;
        this.Cursor = Cursors.Hand;
    }
    private void btnScheduledLessons_MouseLeave(object sender, EventArgs e)
    {
        btnScheduledLessons.BackColor = Color.LightCyan;
        this.Cursor = Cursors.Arrow;
    }
```

```
// Purchased Lessons Button
private void btnPurchasedLessons_MouseEnter(object sender, EventArgs e)
{
    btnPurchasedLessons.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}
private void btnPurchasedLessons_MouseLeave(object sender, EventArgs e)
{
    btnPurchasedLessons.BackColor = Color.LightCyan;
    this.Cursor = Cursors.Arrow;
}

// Login button
private void btnLogin_MouseEnter(object sender, EventArgs e)
{
    btnLogin.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}
private void btnLogin_MouseLeave(object sender, EventArgs e)
{
    btnLogin.BackColor = Color.LightCyan;
    this.Cursor = Cursors.Arrow;
}

private void btnReturn_MouseEnter(object sender, EventArgs e)
{
    btnReturn.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}

private void btnReturn_MouseLeave(object sender, EventArgs e)
{
    btnReturn.BackColor = Color.Firebrick;
    this.Cursor = Cursors.Arrow;
}

private void button3_Click(object sender, EventArgs e)
{
    frmInvoiceStudentSelection Selection = new frmInvoiceStudentSelection();
    Selection.Show();
    this.Hide();
}
```

FrmStudents

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Configuration;
using System.Data.SqlClient;

namespace frmSplash
{
    public partial class frmStudents : Form
    {
        SqlConnection connection;
        string connectionString;
        int MaxStudentID;
        int MinStudentID;
        int CurrentStudentID;
        SqlDataReader DataReader;
        List<int> AllStudentIDs = new List<int>();
        List<string> AllStudent_FirstNames = new List<string>();
        List<string> AllStudent_Surnames = new List<string>();
        List<string> AllStudent_FullNameValues = new List<string>();
        string SearchedStudent;
        char SearchTest;
        string Search_FirstName;
        string Search_Surname;
        List<int> StudentID_FirstNameMatches = new List<int>();
        List<int> StudentID_SurnameMatches = new List<int>();
        int Search_DesiredStudentID;
        bool Firstdisplay;

        public frmStudents()
        {
            InitializeComponent();
            connectionString =
ConfigurationManager.ConnectionStrings["frmSplash.Properties.Settings.PrivateTuitionConnectionString"].ConnectionString;
        }

        private void frmStudents_Load(object sender, EventArgs e)
        {
            DisplayStudent();
        }

        // DISPLAY STUDENT INFORMATION
        public void DisplayStudent()
        {
            using (connection = new SqlConnection(connectionString))
            using (SqlDataAdapter adaptor = new SqlDataAdapter("SELECT * FROM Students", connection))
            {
                connection.Open();

                DataTable StudentTable = new DataTable();
                adaptor.Fill(StudentTable);
            }
        }
    }
}
```

```
lbFirstName.DisplayMember = "First_Name";
lbFirstName.ValueMember = "StudentID";
lbFirstName.DataSource = StudentTable;

lbStudentID.DisplayMember = "StudentID";
lbStudentID.ValueMember = "StudentID";
lbStudentID.DataSource = StudentTable;

lbSurname.DisplayMember = "Surname";
lbSurname.ValueMember = "StudentID";
lbSurname.DataSource = StudentTable;

lbOtherNames.DisplayMember = "Other_Names";
lbOtherNames.ValueMember = "StudentID";
lbOtherNames.DataSource = StudentTable;

lbDOB.DisplayMember = "Date_Of_Birth";
lbDOB.ValueMember = "StudentID";
lbDOB.DataSource = StudentTable;

lbAddress.DisplayMember = "Address";
lbAddress.ValueMember = "StudentID";
lbAddress.DataSource = StudentTable;

lbTown.DisplayMember = "Town";
lbTown.ValueMember = "StudentID";
lbTown.DataSource = StudentTable;

lbPostCode.DisplayMember = "PostCode";
lbPostCode.ValueMember = "StudentID";
lbPostCode.DataSource = StudentTable;

lbContactNum.DisplayMember = "Contact_Number";
lbContactNum.ValueMember = "StudentID";
lbContactNum.DataSource = StudentTable;

lbEmail.DisplayMember = "Email_Address";
lbEmail.ValueMember = "StudentID";
lbEmail.DataSource = StudentTable;

lbGradeID.DisplayMember = "GradeID";
lbGradeID.ValueMember = "StudentID";
lbGradeID.DataSource = StudentTable;

lbInstrumentID.DisplayMember = "InstrumentID";
lbInstrumentID.ValueMember = "StudentID";
lbInstrumentID.DataSource = StudentTable;

lbTuition.DisplayMember = "Tuition_Fee_Recieved";
lbTuition.ValueMember = "StudentID";
lbTuition.DataSource = StudentTable;

connection.Close();

using (SqlCommand Max = new SqlCommand("SELECT MAX(StudentID) FROM
Students", connection))
{
    connection.Open();
    MaxStudentID = (int)Max.ExecuteScalar();
    connection.Close();
}
```

```
        using (SqlCommand Min = new SqlCommand("Select Min(StudentID) FROM Students", connection))
        {
            connection.Open();
            MinStudentID = (int)Min.ExecuteScalar();
            connection.Close();
        }

        StoreAllStudentIDs();
        Fill_StudentSearchBox();

    }

}

// NAVIGATE POSITIVELY THROUGH STUDENT RECORDS
private void NextStudent_Click(object sender, EventArgs e)
{
    CurrentStudentID = Convert.ToInt32(lbStudentID.Text);

    if (CurrentStudentID == MaxStudentID)
    {
        lbStudentID.SelectedValue = MinStudentID;
    }
    else
    {
        CurrentStudentID += 1;
        lbStudentID.SelectedValue = CurrentStudentID;
    }

}

// NAVIGATE NEGATIVELY THROUGH STUDENT RECORDS
private void PreviousStudent_Click(object sender, EventArgs e)
{
    CurrentStudentID = Convert.ToInt32(lbStudentID.Text);

    if (CurrentStudentID == MinStudentID)
    {
        lbStudentID.SelectedValue = MaxStudentID;
    }
    else
    {
        CurrentStudentID += -1;
        lbStudentID.SelectedValue = CurrentStudentID;
    }
}

// RETURN TO MAIN MENU
private void btnReturn_Click(object sender, EventArgs e)
{
    frmPrivateTuition Mainmenu = new frmPrivateTuition();
    Mainmenu.Show();
    this.Hide();
}

// REMOVE STUDENT RECORD
private void btnRemove_Click(object sender, EventArgs e)
{
    // Set associated field.
    GlobalVariables.PreviousForm = "StudentTable";
```

```
// To remove a record, the user needs to login.  
if (GlobalVariables.UserLoggedIn == false)  
{  
    // If no user is currently logged in, load login form.  
    frmUserLogin Login = new frmUserLogin();  
    Login.Show();  
    this.Hide();  
}  
else  
{  
    // If a user is logged in, load confirmation page.  
    GlobalVariables.TableID = Convert.ToInt32(lbStudentID.Text);  
    GlobalVariables.FieldNames = lbFirstName.Text + " " + lbSurname.Text;  
  
    frmConfirmation Confirmation = new frmConfirmation();  
    Confirmation.Show();  
}  
}  
  
// ADD STUDENT FIELD  
private void btnStudentAdd_Click(object sender, EventArgs e)  
{  
    GlobalVariables.PreviousForm = "StudentTable";  
    GlobalVariables.Purpose = "Add";  
  
    frmAddField AddGrade = new frmAddField();  
    AddGrade.Show();  
    this.Hide();  
}  
  
// FIRST AND LAST RECORD DISPLAY  
private void pictureBox5_Click(object sender, EventArgs e)  
{  
    lbStudentID.SelectedValue = AllStudentIDs[0];  
}  
private void pictureBox6_Click(object sender, EventArgs e)  
{  
    lbStudentID.SelectedValue = AllStudentIDs[(AllStudentIDs.Count() - 1)];  
}  
  
// BUTTON MODIFICATIONS  
// Previous Student button  
private void btnPreviousStudent_MouseEnter(object sender, EventArgs e)  
{  
    btnPreviousStudent.BackColor = Color.LightSlateGray;  
    this.Cursor = Cursors.Hand;  
}  
  
private void btnPreviousStudent_MouseLeave(object sender, EventArgs e)  
{  
    btnPreviousStudent.BackColor = Color.MidnightBlue;  
    this.Cursor = Cursors.Arrow;  
}  
  
// Next Student Button  
private void btnNextStudent_MouseEnter(object sender, EventArgs e)  
{  
    btnNextStudent.BackColor = Color.LightSlateGray;  
    this.Cursor = Cursors.Hand;  
}
```

```
private void btnNextStudent_MouseLeave(object sender, EventArgs e)
{
    btnNextStudent.BackColor = Color.MidnightBlue;
    this.Cursor = Cursors.Arrow;
}
// Student Delete button
private void btnRemove_MouseEnter(object sender, EventArgs e)
{
    btnRemove.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}
private void btnRemove_MouseLeave(object sender, EventArgs e)
{
    btnRemove.BackColor = Color.Firebrick;
    this.Cursor = Cursors.Arrow;
}

// Student Add button
private void btnStudentAdd_MouseEnter(object sender, EventArgs e)
{
    btnStudentAdd.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}

private void btnStudentAdd_MouseLeave(object sender, EventArgs e)
{
    btnStudentAdd.BackColor = Color.SeaGreen;
    this.Cursor = Cursors.Arrow;
}

// Close form button
private void btnReturn_MouseEnter(object sender, EventArgs e)
{
    btnReturn.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}

private void btnReturn_MouseLeave(object sender, EventArgs e)
{
    btnReturn.BackColor = Color.Firebrick;
    this.Cursor = Cursors.Arrow;
}

// Update Record
private void btnUpdate_MouseEnter(object sender, EventArgs e)
{
    btnUpdate.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}
private void btnUpdate_MouseLeave(object sender, EventArgs e)
{
    btnUpdate.BackColor = Color.Chocolate;
    this.Cursor = Cursors.Arrow;
}

// First and last buttons
private void pictureBox5_MouseEnter(object sender, EventArgs e)
{
    pictureBox5.Image = Properties.Resources.Arrow2_GreyScale1_;
    this.Cursor = Cursors.Hand;
}
```

```
private void pictureBox5_MouseLeave(object sender, EventArgs e)
{
    pictureBox5.Image = Properties.Resources.Arrow2;
    this.Cursor = Cursors.Arrow;
}

private void pictureBox6_MouseEnter(object sender, EventArgs e)
{
    pictureBox6.Image = Properties.Resources.Arrow2_Inverted__GreyScale1_;
    this.Cursor = Cursors.Hand;
}

private void pictureBox6_MouseLeave(object sender, EventArgs e)
{
    pictureBox6.Image = Properties.Resources.Arrow2_Inverted_;
    this.Cursor = Cursors.Arrow;
}

// STORE STUDENT INFORMATION
public void StoreAllStudentIDs()
{
    // This function will be used to store various pieces of information from
    the schedule Table. This will be used to complete the schedule
    using (connection = new SqlConnection(connectionString))
    {
        connection.Open();
        using (SqlCommand Command = new SqlCommand("SELECT * FROM Students",
connection))
        {
            DataReader = Command.ExecuteReader();
            while (DataReader.Read())
            {
                // For each record, store the corresponding piece of
                information.
                AllStudentIDs.Add(DataReader.GetInt32(0));
                AllStudent_FirstNames.Add(DataReader.GetString(1));
                AllStudent_Surnames.Add(DataReader.GetString(3));
            }
        }
        connection.Close();
    }
}

// FILL SEARCH BOX
public void Fill_StudentSearchBox()
{
    for (int x = 0; x < AllStudentIDs.Count(); x++)
    {
        AllStudent_FullNameValues.Add(AllStudent_FirstNames[x] + " " +
AllStudent_Surnames[x]);
    }

    AllStudent_FullNameValues.Sort();
    cbStudentID.DataSource = AllStudent_FullNameValues;
}

// STUDENT SEARCH FUNCTION
private void cbStudentID_SelectedIndexChanged(object sender, EventArgs e)
{
    StudentID_FirstNameMatches.Clear();
```

```
StudentID_SurnameMatches.Clear();

// identifies and stores selected value.
SearchedStudent = cbStudentID.SelectedValue.ToString();

for (int Y = 0; Y < SearchedStudent.Length; Y++)
{
    // Then take each character present within the string, and separately
look for a 'Space'.
    SearchTest = Convert.ToChar(SearchedStudent.Substring(Y, 1));
    if (SearchTest == ' ')
    {
        // If located, separate first name from surname.
        Search_FirstName = SearchedStudent.Substring(0, (Y));
        Search_Surname = SearchedStudent.Substring((Y + 1),
(SearchedStudent.Length - (Search_FirstName.Length + 1)));
        break;
    }
}

// Then proceed to take the first name, and compare it against all first
names in the system.
for (int x = 0; x < AllStudent_FirstNames.Count(); x++)
{
    // If there is a match, record the associated student ID.
    if (AllStudent_FirstNames[x] == Search_FirstName)
    {
        StudentID_FirstNameMatches.Add(AllStudentIDs[x]);
    }
}

// Then compare the collected surname against the student table
information.
for (int z = 0; z < AllStudent_Surnames.Count(); z++)
{
    // if there is a match, then once again categorise the associated
student ID.
    if (AllStudent_Surnames[z] == Search_Surname)
    {
        StudentID_SurnameMatches.Add(AllStudentIDs[z]);
    }
}

// This block of code will be used to handle the scenario in which
multiple students possess the same first, or surnames.
// I will compare all collected student IDs, for a match should only occur
should the desired student be located.
for (int x = 0; x < StudentID_FirstNameMatches.Count(); x++)
{
    for (int y = 0; y < StudentID_SurnameMatches.Count(); y++)
    {
        if (StudentID_FirstNameMatches[x] == StudentID_SurnameMatches[y])
        {
            // Depending on whether the system contains students with
matching firstnames or surnames, list will contain different values.
            // For this reason, the system will try both possibilities,
should one fail to work.
            try
            {
                Search_DesiredStudentID = StudentID_FirstNameMatches[x];
            }
        }
    }
}
```

```
        catch (Exception SearchFail)
        {
            Search_DesiredStudentID = StudentID_SurnameMatches[y];
        }
    }

}

// Then finally, update the form as to display the information for the
selected student.
lbStudentID.SelectedValue = Search_DesiredStudentID;

if (Firstdisplay == false)
{
    lbStudentID.SelectedValue = AllStudentIDs[0];
    Firstdisplay = true;
}

}

private void btnUpdate_Click(object sender, EventArgs e)
{
    GlobalVariables.PreviousForm = "StudentTable";
    GlobalVariables.Purpose = "Update";
    GlobalVariables.UpdateID =
    lbStudentID.GetItemText(lbStudentID.SelectedItem);

    GlobalVariables.Student_FirstName =
    lbFirstName.GetItemText(lbFirstName.SelectedItem);
    GlobalVariables.Student_OtherNames =
    lbOtherNames.GetItemText(lbOtherNames.SelectedItem);
    GlobalVariables.Student_Surname =
    lbSurname.GetItemText(lbSurname.SelectedItem);
    GlobalVariables.Student_DOB = lbDOB.GetItemText(lbDOB.SelectedItem);
    GlobalVariables.Student_Address =
    lbAddress.GetItemText(lbAddress.SelectedItem);
    GlobalVariables.Student_Town = lbTown.GetItemText(lbTown.SelectedItem);
    GlobalVariables.Student_PostCode =
    lbPostCode.GetItemText(lbPostCode.SelectedItem);
    GlobalVariables.Student_ContactNumber =
    lbContactNum.GetItemText(lbContactNum.SelectedItem);
    GlobalVariables.Student_Email = lbEmail.GetItemText(lbEmail.SelectedItem);
    GlobalVariables.Student_GradeID =
    lbGradeID.GetItemText(lbGradeID.SelectedItem);
    GlobalVariables.Student_InstrumentID =
    lbInstrumentID.GetItemText(lbInstrumentID.SelectedItem);
    GlobalVariables.Student_PaymentFee =
    lbTuition.GetItemText(lbTuition.SelectedItem);

    frmAddField UpdateRecord = new frmAddField();
    UpdateRecord.Show();
    this.Hide();
}

}
```

FrmTeacher

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Configuration;
using System.Data.SqlClient;

namespace frmSplash
{
    public partial class frmTeachers : Form
    {
        SqlConnection connection;
        string connectionString;
        int CurrentTeacherID;
        int MaxTeacherID;
        int MinTeacherID;
        SqlDataReader DataReader;
        List<int> Teacher_IDs = new List<int>();
        List<string> Teachers_FirstName = new List<string>();
        List<string> Teachers_Surname = new List<string>();

        public frmTeachers()
        {
            InitializeComponent();
            connectionString =
ConfigurationManager.ConnectionStrings["frmSplash.Properties.Settings.PrivateTuitionConnectionString"].ConnectionString;
        }

        private void frmTeachers_Load(object sender, EventArgs e)
        {
            DisplayTeachers();
        }

        // DISPLAY TEACHER INFORMATION
        public void DisplayTeachers()
        {
            using (connection = new SqlConnection(connectionString))
            {
                using (SqlDataAdapter adaptor = new SqlDataAdapter("SELECT * FROM Teachers", connection))
                {
                    DataTable TeacherTable = new DataTable();
                    adaptor.Fill(TeacherTable);

                    cbTeacherID.DisplayMember = "First_Name";
                    cbTeacherID.ValueMember = "TeacherID";
                    cbTeacherID.DataSource = TeacherTable;

                    lbTeacherID.DisplayMember = "TeacherID";
                    lbTeacherID.ValueMember = "TeacherID";
                    lbTeacherID.DataSource = TeacherTable;

                    lbFirstName.DisplayMember = "First_Name";
                }
            }
        }
    }
}
```

```
lbFirstName.ValueMember = "TeacherID";
lbFirstName.DataSource = TeacherTable;

lbSurname.DisplayMember = "Surname";
lbSurname.ValueMember = "TeacherID";
lbSurname.DataSource = TeacherTable;

lbAddress.DisplayMember = "Address";
lbAddress.ValueMember = "TeacherID";
lbAddress.DataSource = TeacherTable;

lbTown.DisplayMember = "Town";
lbTown.ValueMember = "TeacherID";
lbTown.DataSource = TeacherTable;

lbPostCode.DisplayMember = "PostCode";
lbPostCode.ValueMember = "TeacherID";
lbPostCode.DataSource = TeacherTable;

lbContactNum.DisplayMember = "Contact_Number";
lbContactNum.ValueMember = "TeacherID";
lbContactNum.DataSource = TeacherTable;

lbEmail.DisplayMember = "Email_Address";
lbEmail.ValueMember = "TeacherID";
lbEmail.DataSource = TeacherTable;

lbSpecialisation.DisplayMember = "Specialisation";
lbSpecialisation.ValueMember = "TeacherID";
lbSpecialisation.DataSource = TeacherTable;

lbRoomID.DisplayMember = "RoomID";
lbRoomID.ValueMember = "TeacherID";
lbRoomID.DataSource = TeacherTable;

    using (SqlCommand Max = new SqlCommand("SELECT MAX(TeacherID) FROM
Teachers", connection))
    {
        connection.Open();
        MaxTeacherID = (int)Max.ExecuteScalar();
    }

    using (SqlCommand Min = new SqlCommand("Select Min(TeacherID) FROM
Teachers", connection))
    {
        MinTeacherID = (int)Min.ExecuteScalar();
        connection.Close();
    }
}

using (connection = new SqlConnection(connectionString))
{
    connection.Open();
    using (SqlCommand Command = new SqlCommand("SELECT * FROM Teachers",
connection))
    {
        DataReader = Command.ExecuteReader();
        while (DataReader.Read())
        {
```

```
// For each record, store the corresponding piece of
information.
        Teacher_IDs.Add(DataReader.GetInt32(0));
        Teachers_FirstName.Add(DataReader.GetString(1));
        Teachers_Surname.Add(DataReader.GetString(2));
    }
}
connection.Close();
}

// RETURN TO MAIN MENU
private void btnReturn_Click(object sender, EventArgs e)
{
    frmPrivateTuition Mainmenu = new frmPrivateTuition();
    Mainmenu.Show();
    this.Hide();
}

// VIEW PREVIOUS TEACHER RECORD
private void btnPreviousStudent_Click(object sender, EventArgs e)
{
    CurrentTeacherID = Convert.ToInt32(lbTeacherID.SelectedValue);

    if (CurrentTeacherID == MinTeacherID)
    {
        lbTeacherID.SelectedValue = MaxTeacherID;
    }
    else
    {
        CurrentTeacherID += -1;
        lbTeacherID.SelectedValue = CurrentTeacherID;
    }
}

// VIEW NEXT TEACHER RECORD
private void btnNextStudent_Click(object sender, EventArgs e)
{
    CurrentTeacherID = Convert.ToInt32(lbTeacherID.SelectedValue);

    if (CurrentTeacherID == MaxTeacherID)
    {
        lbTeacherID.SelectedValue = MinTeacherID;
    }
    else
    {
        CurrentTeacherID += 1;
        lbTeacherID.SelectedValue = CurrentTeacherID;
    }
}

// ADD NEW RECORD
private void btnAdd_Click(object sender, EventArgs e)
{
    GlobalVariables.PreviousForm = "TeacherTable";

    frmAddField AddTeacher = new frmAddField();
    AddTeacher.Show();
    this.Hide();
}

// REMOVE RECORD
```

```
private void btnRemove_Click(object sender, EventArgs e)
{
    // Set associated field.
    GlobalVariables.PreviousForm = "TeacherTable";

    // To remove a record, the user needs to login.
    if (GlobalVariables.UserLoggedIn == false)
    {
        // If no user is currently logged in, load login form.
        frmUserLogin Login = new frmUserLogin();
        Login.Show();
        this.Hide();
    }
    else
    {
        // If a user is logged in, load confirmation page.
        GlobalVariables.TableID = Convert.ToInt32(lbTeacherID.Text);
        GlobalVariables.FieldNames = lbFirstName.Text + " " + lbSurname.Text;

        frmConfirmation Confirmation = new frmConfirmation();
        Confirmation.Show();
    }
}

// BUTTON SELECT MODIFICATIONS
// Previous Record
private void btnPreviousTeacher_MouseEnter(object sender, EventArgs e)
{
    btnPreviousTeacher.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}
private void btnPreviousTeacher_MouseLeave(object sender, EventArgs e)
{
    btnPreviousTeacher.BackColor = Color.MidnightBlue;
    this.Cursor = Cursors.Arrow;
}

// Next Record
private void btnNextTeacher_MouseEnter(object sender, EventArgs e)
{
    btnNextTeacher.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}
private void btnNextTeacher_MouseLeave(object sender, EventArgs e)
{
    btnNextTeacher.BackColor = Color.MidnightBlue;
    this.Cursor = Cursors.Arrow;
}

// Add Record
private void btnAdd_MouseEnter(object sender, EventArgs e)
{
    btnAdd.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}
private void btnAdd_MouseLeave(object sender, EventArgs e)
{
    btnAdd.BackColor = Color.SeaGreen;
    this.Cursor = Cursors.Arrow;
}
```

```
        // Remove Record
private void btnRemove_MouseEnter(object sender, EventArgs e)
{
    btnRemove.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}
private void btnRemove_MouseLeave(object sender, EventArgs e)
{
    btnRemove.BackColor = Color.Firebrick;
    this.Cursor = Cursors.Arrow;
}

        // Return To Main Menu
private void btnReturn_MouseEnter(object sender, EventArgs e)
{
    btnReturn.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}
private void btnReturn_MouseLeave(object sender, EventArgs e)
{
    btnReturn.BackColor = Color.Firebrick;
    this.Cursor = Cursors.Arrow;
}

        // First Record
private void pictureBox5_MouseEnter(object sender, EventArgs e)
{
    pbFirstRecord.Image = Properties.Resources.Arrow2_GreyScale1_;
    this.Cursor = Cursors.Hand;
}
private void pictureBox5_MouseLeave(object sender, EventArgs e)
{
    pbFirstRecord.Image = Properties.Resources.Arrow2;
    this.Cursor = Cursors.Arrow;
}

        // Last Record
private void pbLastRecord_MouseEnter(object sender, EventArgs e)
{
    pbLastRecord.Image = Properties.Resources.Arrow2_Inverted_GreyScale1_;
    this.Cursor = Cursors.Hand;
}
private void pbLastRecord_MouseLeave(object sender, EventArgs e)
{
    pbLastRecord.Image = Properties.Resources.Arrow2_Inverted_;
    this.Cursor = Cursors.Arrow;
}

        // Update Record
private void btnUpdate_MouseEnter(object sender, EventArgs e)
{
    btnUpdate.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}
private void btnUpdate_MouseLeave(object sender, EventArgs e)
{
    btnUpdate.BackColor = Color.Chocolate;
    this.Cursor = Cursors.Arrow;
}

private void btnUpdate_Click(object sender, EventArgs e)
```

```
        GlobalVariables.PreviousForm = "TeacherTable";
        GlobalVariables.Purpose = "Update";
        GlobalVariables.UpdateID =
lbTeacherID.GetItemText(lbTeacherID.SelectedValue);

        GlobalVariables.Teacher_FirstName =
lbFirstName.GetItemText(lbFirstName.SelectedItem);
        GlobalVariables.Teacher_Surname =
lbSurname.GetItemText(lbSurname.SelectedItem);
        GlobalVariables.Teacher_Address =
lbAddress.GetItemText(lbAddress.SelectedItem);
        GlobalVariables.Teacher_Town = lbTown.GetItemText(lbTown.SelectedItem);
        GlobalVariables.Teacher_PostCode =
lbPostCode.GetItemText(lbPostCode.SelectedItem);
        GlobalVariables.Teacher_ContactNumber =
lbContactNum.GetItemText(lbContactNum.SelectedItem);
        GlobalVariables.Teacher_Email = lbEmail.GetItemText(lbEmail.SelectedItem);
        GlobalVariables.Teacher_Specialisation =
lbSpecialisation.GetItemText(lbSpecialisation.SelectedItem);
        GlobalVariables.Teacher_RoomID =
lbRoomID.GetItemText(lbRoomID.SelectedItem);

        frmAddField UpdateRecord = new frmAddField();
        UpdateRecord.Show();
        this.Hide();
    }

    private void pbLastRecord_Click(object sender, EventArgs e)
{
    lbTeacherID.SelectedValue = Teacher_IDs[(Teacher_IDs.Count() - 1)];
}

    private void pbFirstRecord_Click(object sender, EventArgs e)
{
    lbTeacherID.SelectedValue = Teacher_IDs[0];
}
}
```

FrmInstrument

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Configuration;
using System.Data.SqlClient;

namespace frmSplash
{
    public partial class frmInstrument : Form
    {
        SqlConnection connection;
        string connectionString;
        int MaxInstruments;
        int MinInstruments;
        int currentInstrumentID;
        SqlDataReader DataReader;
        List<int> Instrument_IDs = new List<int>();

        public frmInstrument()
        {
            InitializeComponent();
            connectionString =
ConfigurationManager.ConnectionStrings["frmSplash.Properties.Settings.PrivateTuitionCo
nectionString"].ConnectionString;
        }

        private void frmInstrument_Load(object sender, EventArgs e)
        {
            DisplayInstruments();
        }

        // STORE INSTRUMENT DETIALS
        public void DisplayInstruments()
        {
            using (connection = new SqlConnection(connectionString))
            {
                using (SqlDataAdapter adaptor = new SqlDataAdapter("SELECT * FROM
Instruments", connection))
                {
                    DataTable InstrumentTable = new DataTable();
                    adaptor.Fill(InstrumentTable);

                    cbInstrumentSearch.DisplayMember = "Instrument Name";
                    cbInstrumentSearch.ValueMember = "InstrumentID";
                    cbInstrumentSearch.DataSource = InstrumentTable;

                    lbInstrumentID.DisplayMember = "InstrumentID";
                    lbInstrumentID.ValueMember = "InstrumentID";
                    lbInstrumentID.DataSource = InstrumentTable;

                    lbInstrumentType.DisplayMember = "Instrument Type";
                    lbInstrumentType.ValueMember = "InstrumentID";
                    lbInstrumentType.DataSource = InstrumentTable;
                }
            }
        }
    }
}
```

```
        lbInstrumentName.DisplayMember = "Instrument Name";
        lbInstrumentName.ValueMember = "InstrumentID";
        lbInstrumentName.DataSource = InstrumentTable;

        lbQuantity.DisplayMember = "Quantity";
        lbQuantity.ValueMember = "InstrumentID";
        lbQuantity.DataSource = InstrumentTable;

        using (SqlCommand Max = new SqlCommand("SELECT MAX(InstrumentID)
FROM Instruments", connection))
        {
            connection.Open();
            MaxInstruments = (int)Max.ExecuteScalar();
        }

        using (SqlCommand Min = new SqlCommand("Select Min(InstrumentID)
FROM Instruments", connection))
        {
            MinInstruments = (int)Min.ExecuteScalar();
            connection.Close();
        }
    }

    using (connection = new SqlConnection(connectionString))
    {
        connection.Open();
        using (SqlCommand Command = new SqlCommand("SELECT * FROM
Instruments", connection))
        {
            DataReader = Command.ExecuteReader();
            while (DataReader.Read())
            {
                // For each record, store the corrisponding peice of
information.
                Instrument_IDs.Add(DataReader.GetInt32(0));
            }
        }
        connection.Close();
    }

    // VIEW PREVIOUS INSTRUMENT
    private void btnPreviousStudent_Click(object sender, EventArgs e)
    {
        currentInstrumentID = Convert.ToInt32(lbInstrumentID.SelectedValue);

        if (currentInstrumentID == MinInstruments)
        {
            lbInstrumentID.SelectedValue = MaxInstruments;
        }
        else
        {
            currentInstrumentID += -1;
            lbInstrumentID.SelectedValue = currentInstrumentID;
        }
    }

    // VIEW NEXT INSTRUMENT
    private void btnNextStudent_Click(object sender, EventArgs e)
    {
```

```
currentInstrumentID = Convert.ToInt32(lbInstrumentID.SelectedValue);

if (currentInstrumentID == MaxInstruments)
{
    lbInstrumentID.SelectedValue = MinInstruments;
}
else
{
    currentInstrumentID += 1;
    lbInstrumentID.SelectedValue = currentInstrumentID;
}

// RETURN TO MAIN MENU
private void btnReturn_Click(object sender, EventArgs e)
{
    frmPrivateTuition Mainmenu = new frmPrivateTuition();
    Mainmenu.Show();
    this.Hide();
}

// REMOVE INSTRUMENT
private void btnRemove_Click(object sender, EventArgs e)
{
    // Set associated field.
    GlobalVariables.PreviousForm = "InstrumentTable";

    // To remove a record, the user needs to login.
    if (GlobalVariables.UserLoggedIn == false)
    {
        // If no user is currently logged in, load login form.
        frmUserLogin Login = new frmUserLogin();
        Login.Show();
        this.Hide();
    }
    else
    {
        // If a user is logged in, load confirmation page.
        GlobalVariables.TableID = Convert.ToInt32(lbInstrumentID.Text);
        GlobalVariables.FieldName = lbInstrumentName.Text;

        frmConfirmation Confirmation = new frmConfirmation();
        Confirmation.Show();
    }
}

// ADD NEW INSTRUMENT
private void btnAddGrade_Click(object sender, EventArgs e)
{
    GlobalVariables.PreviousForm = "InstrumentTable";
    GlobalVariables.Purpose = "Add";

    frmAddField AddField = new frmAddField();
    AddField.Show();
    this.Hide();
}

// BUTTON SELECTED MODIFICATIONS
// Previous Instrument
private void btnPreviousInstrument_MouseEnter(object sender, EventArgs e)
{
    btnPreviousInstrument.BackColor = Color.LightSlateGray;
```

```
        this.Cursor = Cursors.Hand;
    }

    private void btnPreviousInstrument_MouseLeave(object sender, EventArgs e)
{
    btnPreviousInstrument.BackColor = Color.MidnightBlue;
    this.Cursor = Cursors.Arrow;
}

// Next Instrument
private void btnNextInstrument_MouseEnter(object sender, EventArgs e)
{
    btnNextInstrument.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}

private void btnNextInstrument_MouseLeave(object sender, EventArgs e)
{
    btnNextInstrument.BackColor = Color.MidnightBlue;
    this.Cursor = Cursors.Arrow;
}

// Add new Instrument
private void btnAddGrade_MouseEnter(object sender, EventArgs e)
{
    btnAddInstrument.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}

private void btnAddGrade_MouseLeave(object sender, EventArgs e)
{
    btnAddInstrument.BackColor = Color.SeaGreen;
    this.Cursor = Cursors.Arrow;
}

// Remove Instrument
private void btnRemove_MouseEnter(object sender, EventArgs e)
{
    btnRemove.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}

private void btnRemove_MouseLeave(object sender, EventArgs e)
{
    btnRemove.BackColor = Color.Firebrick;
    this.Cursor = Cursors.Arrow;
}

// Return to Main Menu
private void btnReturn_MouseEnter(object sender, EventArgs e)
{
    btnReturn.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}

private void btnReturn_MouseLeave(object sender, EventArgs e)
{
    btnReturn.BackColor = Color.Firebrick;
    this.Cursor = Cursors.Arrow;
}

// First Record
```

```
private void pictureBox5_MouseEnter(object sender, EventArgs e)
{
    pbFirstRecord.Image = Properties.Resources.Arrow2_GreyScale1_;
    this.Cursor = Cursors.Hand;
}

private void pictureBox5_MouseLeave(object sender, EventArgs e)
{
    pbFirstRecord.Image = Properties.Resources.Arrow2;
    this.Cursor = Cursors.Arrow;
}

// Last Record
private void pictureBox6_MouseEnter(object sender, EventArgs e)
{
    pbLastRecord.Image = Properties.Resources.Arrow2_Inverted_GreyScale1_;
    this.Cursor = Cursors.Hand;
}

private void pictureBox6_MouseLeave(object sender, EventArgs e)
{
    pbLastRecord.Image = Properties.Resources.Arrow2_Inverted_;
    this.Cursor = Cursors.Arrow;
}

private void btnUpdate_Click(object sender, EventArgs e)
{
    GlobalVariables.PreviousForm = "InstrumentTable";
    GlobalVariables.Purpose = "Update";
    GlobalVariables.UpdateID =
    lbInstrumentID.GetItemText(lbInstrumentID.SelectedItem);

    GlobalVariables.Instrument_Type =
    lbInstrumentType.GetItemText(lbInstrumentType.SelectedItem);
    GlobalVariables.Instrument_Name =
    lbInstrumentName.GetItemText(lbInstrumentName.SelectedItem);
    GlobalVariables.Instrument_Quantity =
    lbQuantity.GetItemText(lbQuantity.SelectedItem);

    frmAddField UpdateRecord = new frmAddField();
    UpdateRecord.Show();
    this.Hide();
}

// FIRST / LAST RECORD
private void pbFirstRecord_Click(object sender, EventArgs e)
{
    lbInstrumentID.SelectedValue = Instrument_IDs[0];
}
private void pbLastRecord_Click(object sender, EventArgs e)
{
    lbInstrumentID.SelectedValue = Instrument_IDs[(Instrument_IDs.Count() -
1)];
}

// UPDATE BUTTON
private void btnUpdate_MouseEnter(object sender, EventArgs e)
{
    btnUpdate.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}
private void btnUpdate_MouseLeave(object sender, EventArgs e)
```

```
{  
    btnUpdate.BackColor = Color.Chocolate;  
    this.Cursor = Cursors.Arrow;  
}  
}  
}
```

FrmGrade

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Configuration;
using System.Data.SqlClient;

namespace frmSplash
{
    public partial class frmGrade : Form
    {
        SqlConnection connection;
        string connectionString;
        int CurrentGradeID;
        int MaxGradeID;
        int MinGradeID;
        SqlDataReader DataReader;

        List<int> Grade_IDs = new List<int>();

        public frmGrade()
        {
            InitializeComponent();
            connectionString =
ConfigurationManager.ConnectionStrings["frmSplash.Properties.Settings.PrivateTuitionCo
nnectionString"].ConnectionString;
        }

        // UPDATE INFORMATION DISPLAYS
        public void Display_GradeDetails()
        {
            using (connection = new SqlConnection(connectionString))
            {
                using (SqlDataAdapter adaptor = new SqlDataAdapter("SELECT * FROM
Grade", connection))
                {
                    DataTable GradeTable = new DataTable();
                    adaptor.Fill(GradeTable);

                    cbGradeSearch.DisplayMember = "GradeLevel";
                    cbGradeSearch.ValueMember = "GradeID";
                    cbGradeSearch.DataSource = GradeTable;

                    lbGradeID.DisplayMember = "GradeID";
                    lbGradeID.ValueMember = "GradeID";
                    lbGradeID.DataSource = GradeTable;

                    lbGradeLevel.DisplayMember = "GradeLevel";
                    lbGradeLevel.ValueMember = "GradeID";
                    lbGradeLevel.DataSource = GradeTable;

                    lbGradeFee.DisplayMember = "Grade Fee";
                    lbGradeFee.ValueMember = "GradeID";
                    lbGradeFee.DataSource = GradeTable;
                }
            }
        }
    }
}
```

```
        using (SqlCommand Max = new SqlCommand("SELECT MAX(GradeID) FROM Grade", connection))
    {
        connection.Open();
        MaxGradeID = (int)Max.ExecuteScalar();
    }

    using (SqlCommand Min = new SqlCommand("Select MIN(GradeID) FROM Grade", connection))
    {
        MinGradeID = (int)Min.ExecuteScalar();
        connection.Close();
    }
}

using (connection = new SqlConnection(connectionString))
{
    connection.Open();
    using (SqlCommand Command = new SqlCommand("SELECT * FROM Grade", connection))
    {
        DataReader = Command.ExecuteReader();
        while (DataReader.Read())
        {
            // For each record, store the corresponding piece of information.
            Grade_IDs.Add(DataReader.GetInt32(0));
        }
    }
    connection.Close();
}

private void frmGrade_Load(object sender, EventArgs e)
{
    Display_GradeDetails();
}

// VIEW PREVIOUS GRADE RECORD
private void btnPreviousGrade_Click(object sender, EventArgs e)
{
    CurrentGradeID = Convert.ToInt32(lbGradeID.SelectedValue);

    if (CurrentGradeID == MinGradeID)
    {
        lbGradeID.SelectedValue = MaxGradeID;
    }
    else
    {
        CurrentGradeID += -1;
        lbGradeID.SelectedValue = CurrentGradeID;
    }
}

// VIEW NEXT GRADE RECORD
private void btnNextGrade_Click(object sender, EventArgs e)
{
    CurrentGradeID = Convert.ToInt32(lbGradeID.SelectedValue);

    if (CurrentGradeID == MaxGradeID)
```

```
{  
    lbGradeID.SelectedValue = MinGradeID;  
}  
else  
{  
    CurrentGradeID += 1;  
    lbGradeID.SelectedValue = CurrentGradeID;  
}  
}  
  
// RETURN TO MAIN MENU  
private void btnReturn_Click(object sender, EventArgs e)  
{  
    frmPrivateTuition MainMenu = new frmPrivateTuition();  
    MainMenu.Show();  
    this.Hide();  
}  
  
// ADD GRADE RECORD  
private void btnAddGrade_Click(object sender, EventArgs e)  
{  
    GlobalVariables.PreviousForm = "GradeTable";  
    GlobalVariables.Purpose = "Add";  
  
    frmAddField AddGrade = new frmAddField();  
    AddGrade.Show();  
    this.Hide();  
}  
  
// REMOVE GRADE RECORD  
private void btnRemove_Click(object sender, EventArgs e)  
{  
    // Set associated field.  
    GlobalVariables.PreviousForm = "GradeTable";  
  
    // To remove a record, the user needs to login.  
    if (GlobalVariables.UserLoggedIn == false)  
    {  
        // If no user is currently logged in, load login form.  
        frmUserLogin Login = new frmUserLogin();  
        Login.Show();  
        this.Hide();  
    }  
    else  
    {  
        // If a user is logged in, load confirmation page.  
        GlobalVariables.TableID = Convert.ToInt32(lbGradeID.Text);  
        GlobalVariables.FieldNames = lbGradeLevel.Text;  
  
        frmConfirmation Confirmation = new frmConfirmation();  
        Confirmation.Show();  
    }  
}  
  
// BUTTON SELECT FEEDBACK  
// Previous grade  
private void btnPreviousGrade_MouseEnter(object sender, EventArgs e)  
{  
    btnPreviousGrade.BackColor = Color.LightSlateGray;  
    this.Cursor = Cursors.Hand;  
}
```

```
private void btnPreviousGrade_MouseLeave(object sender, EventArgs e)
{
    btnPreviousGrade.BackColor = Color.MidnightBlue;
    this.Cursor = Cursors.Arrow;
}

// Next Grade
private void btnNextGrade_MouseEnter(object sender, EventArgs e)
{
    btnNextGrade.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}

private void btnNextGrade_MouseLeave(object sender, EventArgs e)
{
    btnNextGrade.BackColor = Color.MidnightBlue;
    this.Cursor = Cursors.Arrow;
}

// Add Grade
private void btnAddGrade_MouseEnter(object sender, EventArgs e)
{
    btnAddGrade.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}

private void btnAddGrade_MouseLeave(object sender, EventArgs e)
{
    btnAddGrade.BackColor = Color.SeaGreen;
    this.Cursor = Cursors.Arrow;
}

// Remove Grade
private void btnRemove_MouseEnter(object sender, EventArgs e)
{
    btnRemove.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}

private void btnRemove_MouseLeave(object sender, EventArgs e)
{
    btnRemove.BackColor = Color.Firebrick;
    this.Cursor = Cursors.Arrow;
}

// Return to Main Menu
private void btnReturn_MouseEnter(object sender, EventArgs e)
{
    btnReturn.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}

private void btnReturn_MouseLeave(object sender, EventArgs e)
{
    btnReturn.BackColor = Color.Firebrick;
    this.Cursor = Cursors.Arrow;
}

// First Grade
private void pictureBox5_MouseEnter(object sender, EventArgs e)
{
```

```
        pbFirstRecord.Image = Properties.Resources.Arrow2_GreyScale1_;
        this.Cursor = Cursors.Hand;
    }

    private void pictureBox5_MouseLeave(object sender, EventArgs e)
    {
        pbFirstRecord.Image = Properties.Resources.Arrow2;
        this.Cursor = Cursors.Arrow;
    }

    // Last Grade
    private void pictureBox6_MouseEnter(object sender, EventArgs e)
    {
        pbLastRecord.Image = Properties.Resources.Arrow2_Inverted_GreyScale1_;
        this.Cursor = Cursors.Hand;
    }

    private void pictureBox6_MouseLeave(object sender, EventArgs e)
    {
        pbLastRecord.Image = Properties.Resources.Arrow2_Inverted_;
        this.Cursor = Cursors.Arrow;
    }

    // Update Record
    private void btnUpdate_MouseEnter(object sender, EventArgs e)
    {
        btnUpdate.BackColor = Color.LightSlateGray;
        this.Cursor = Cursors.Hand;
    }
    private void btnUpdate_MouseLeave(object sender, EventArgs e)
    {
        btnUpdate.BackColor = Color.Chocolate;
        this.Cursor = Cursors.Arrow;
    }

    // UPDATE RECORD
    private void btnUpdate_Click(object sender, EventArgs e)
    {
        GlobalVariables.PreviousForm = "GradeTable";
        GlobalVariables.Purpose = "Update";
        GlobalVariables.UpdateID = lbGradeID.GetItemText(lbGradeID.SelectedItem);

        GlobalVariables.Grade_Level =
        lbGradeLevel.GetItemText(lbGradeLevel.SelectedItem);
        GlobalVariables.Grade_Fee =
        lbGradeFee.GetItemText(lbGradeFee.SelectedItem);

        frmAddField UpdateRecord = new frmAddField();
        UpdateRecord.Show();
        this.Hide();
    }

    // FIRST / LAST RECORD
    private void pbFirstRecord_Click(object sender, EventArgs e)
    {
        lbGradeID.SelectedValue = Grade_IDs[0];
    }
    private void pbLastRecord_Click(object sender, EventArgs e)
    {
        lbGradeID.SelectedValue = Grade_IDs[(Grade_IDs.Count() - 1)];
    }
}
```

```
        }  
    }  
}
```

FrmRoom

```
using System;  
using System.Collections.Generic;  
using System.ComponentModel;  
using System.Data;  
using System.Drawing;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
using System.Windows.Forms;  
using System.Configuration;  
using System.Data.SqlClient;  
  
namespace frmSplash  
{  
    public partial class frmRoom : Form  
    {  
        string connectionString;  
        SqlConnection connection;  
        int CurrentRoomID;  
        int MaxRoomID;  
        int MinRoomID;  
        SqlDataReader DataReader;  
        List<int> Room_IDs = new List<int>();  
  
        public frmRoom()  
        {  
            InitializeComponent();  
            connectionString =  
ConfigurationManager.ConnectionStrings["frmSplash.Properties.Settings.PrivateTuitionCo  
nnnectionString"].ConnectionString;  
        }  
  
        private void frmRoom_Load(object sender, EventArgs e)  
        {  
            DisplayRooms();  
        }  
  
        // DISPLAY TABLE INFORMATION  
        public void DisplayRooms()  
        {  
            using (connection = new SqlConnection(connectionString))  
            {  
                using (SqlDataAdapter adaptor = new SqlDataAdapter("SELECT * FROM  
Room", connection))  
                {  
                    DataTable RoomTable = new DataTable();  
                    adaptor.Fill(RoomTable);  
  
                    cbRoomID.DisplayMember = "RoomID";  
                    cbRoomID.ValueMember = "RoomID";  
                    cbRoomID.DataSource = RoomTable;  
  
                    lbRoomID.DisplayMember = "RoomID";  
                    lbRoomID.ValueMember = "RoomID";  
                }  
            }  
        }  
}
```

```
lbRoomID.DataSource = RoomTable;

lbRoomType.DisplayMember = "Room_Type";
lbRoomType.ValueMember = "RoomID";
lbRoomType.DataSource = RoomTable;

lbRoomSpecialisation.DisplayMember = "specialisation";
lbRoomSpecialisation.ValueMember = "RoomID";
lbRoomSpecialisation.DataSource = RoomTable;

using (SqlCommand Max = new SqlCommand("SELECT MAX(RoomID) FROM
Room", connection))
{
    connection.Open();
    MaxRoomID = (int)Max.ExecuteScalar();
}

using (SqlCommand Min = new SqlCommand("Select Min(RoomID) FROM
Room", connection))
{
    MinRoomID = (int)Min.ExecuteScalar();
    connection.Close();
}
}

using (connection = new SqlConnection(connectionString))
{
    connection.Open();
    using (SqlCommand Command = new SqlCommand("SELECT * FROM Room",
connection))
    {
        DataReader = Command.ExecuteReader();
        while (DataReader.Read())
        {
            // For each record, store the corrisponding peice of
information.
            Room_IDs.Add(DataReader.GetInt32(0));
        }
    }
    connection.Close();
}

// VIEW PREVIOUS RECORD
private void btnPreviousRoom_Click(object sender, EventArgs e)
{
    CurrentRoomID = Convert.ToInt32(lbRoomID.SelectedValue);

    if (CurrentRoomID == MinRoomID)
    {
        lbRoomID.SelectedValue = MaxRoomID;
    }
    else
    {
        CurrentRoomID += -1;
        lbRoomID.SelectedValue = CurrentRoomID;
    }
}

// VIEW NEXT RECORD
private void btnNextRoom_Click(object sender, EventArgs e)
{
```

```
CurrentRoomID = Convert.ToInt32(lbRoomID.SelectedValue);

if (CurrentRoomID == MaxRoomID)
{
    lbRoomID.SelectedValue = MinRoomID;
}
else
{
    CurrentRoomID += 1;
    lbRoomID.SelectedValue = CurrentRoomID;
}

// RETURN TO MAIN MENU
private void btnReturn_Click(object sender, EventArgs e)
{
    frmPrivateTuition Mainmenu = new frmPrivateTuition();
    Mainmenu.Show();
    this.Hide();
}

// ADD RECORD
private void btnAddRoom_Click(object sender, EventArgs e)
{
    GlobalVariables.PreviousForm = "RoomTable";
    GlobalVariables.Purpose = "Add";

    frmAddField AddRoom = new frmAddField();
    AddRoom.Show();
    this.Hide();
}

// REMOVE RECORD
private void btnRemove_Click(object sender, EventArgs e)
{
    // Set associated field.
    GlobalVariables.PreviousForm = "RoomTable";

    // To remove a record, the user needs to login.
    if (GlobalVariables.UserLoggedIn == false)
    {
        // If no user is currently logged in, load login form.
        frmUserLogin Login = new frmUserLogin();
        Login.Show();
        this.Hide();
    }
    else
    {
        // If a user is logged in, load confirmation page.
        GlobalVariables.TableID = Convert.ToInt32(lbRoomID.Text);
        GlobalVariables.FieldNames = lbRoomSpecialisation.Text;

        frmConfirmation Confirmation = new frmConfirmation();
        Confirmation.Show();
    }
}

// BUTTON SELECT MODIFICATIONS
// Previous Record
private void btnPreviousRoom_MouseEnter(object sender, EventArgs e)
{
    btnPreviousRoom.BackColor = Color.LightSlateGray;
```

```
        this.Cursor = Cursors.Hand;
    }
    private void btnPreviousRoom_MouseLeave(object sender, EventArgs e)
    {
        btnPreviousRoom.BackColor = Color.MidnightBlue;
        this.Cursor = Cursors.Arrow;
    }

    // Next Record
    private void btnNextRoom_MouseEnter(object sender, EventArgs e)
    {
        btnNextRoom.BackColor = Color.LightSlateGray;
        this.Cursor = Cursors.Hand;
    }
    private void btnNextRoom_MouseLeave(object sender, EventArgs e)
    {
        btnNextRoom.BackColor = Color.MidnightBlue;
        this.Cursor = Cursors.Arrow;
    }

    // Add New Record
    private void btnAddRoom_MouseEnter(object sender, EventArgs e)
    {
        btnAddRoom.BackColor = Color.LightSlateGray;
        this.Cursor = Cursors.Hand;
    }
    private void btnAddRoom_MouseLeave(object sender, EventArgs e)
    {
        btnAddRoom.BackColor = Color.SeaGreen;
        this.Cursor = Cursors.Arrow;
    }

    // Remove Record
    private void btnRemove_MouseEnter(object sender, EventArgs e)
    {
        btnRemove.BackColor = Color.LightSlateGray;
        this.Cursor = Cursors.Hand;
    }
    private void btnRemove_MouseLeave(object sender, EventArgs e)
    {
        btnRemove.BackColor = Color.Firebrick;
        this.Cursor = Cursors.Arrow;
    }

    // Return to Main Menu
    private void btnReturn_MouseEnter(object sender, EventArgs e)
    {
        btnReturn.BackColor = Color.LightSlateGray;
        this.Cursor = Cursors.Hand;
    }
    private void btnReturn_MouseLeave(object sender, EventArgs e)
    {
        btnReturn.BackColor = Color.Firebrick;
        this.Cursor = Cursors.Arrow;
    }

    // First Record
    private void pbFirstRecord_MouseEnter(object sender, EventArgs e)
    {
        pbFirstRecord.Image = Properties.Resources.Arrow2_GreyScale1_;
        this.Cursor = Cursors.Hand;
    }
```

```
private void pbFirstRecord_MouseLeave(object sender, EventArgs e)
{
    pbFirstRecord.Image = Properties.Resources.Arrow2;
    this.Cursor = Cursors.Arrow;
}

// Last Record
private void pbLastRecord_MouseEnter(object sender, EventArgs e)
{
    pbLastRecord.Image = Properties.Resources.Arrow2_Inverted_GreyScale1_;
    this.Cursor = Cursors.Hand;
}
private void pbLastRecord_MouseLeave(object sender, EventArgs e)
{
    pbLastRecord.Image = Properties.Resources.Arrow2_Inverted_;
    this.Cursor = Cursors.Arrow;
}

// Update Record
private void btnUpdate_MouseEnter(object sender, EventArgs e)
{
    btnUpdate.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}
private void btnUpdate_MouseLeave(object sender, EventArgs e)
{
    btnUpdate.BackColor = Color.Chocolate;
    this.Cursor = Cursors.Arrow;
}

private void btnUpdate_Click(object sender, EventArgs e)
{
    GlobalVariables.PreviousForm = "RoomTable";
    GlobalVariables.Purpose = "Update";
    GlobalVariables.UpdateID = lbRoomID.GetItemText(lbRoomID.SelectedItem);

    GlobalVariables.Room_Type =
    lbRoomType.GetItemText(lbRoomType.SelectedItem);
    GlobalVariables.Room_Specialisation =
    lbRoomSpecialisation.GetItemText(lbRoomSpecialisation.SelectedItem);

    frmAddField UpdateRecord = new frmAddField();
    UpdateRecord.Show();
    this.Hide();
}

// FIRST / LAST RECORD
private void pbFirstRecord_Click(object sender, EventArgs e)
{
    lbRoomID.SelectedValue = Room_IDs[0];
}
private void pbLastRecord_Click(object sender, EventArgs e)
{
    lbRoomID.SelectedValue = Room_IDs[(Room_IDs.Count() - 1)];
}
}
```

FrmLessonBundle

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Configuration;
using System.Data.SqlClient;

namespace frmSplash
{
    public partial class frmLessonBundle : Form
    {
        SqlConnection connection;
        string connectionString;
        int MaxBundleID;
        int MinBundleID;
        int CurrentBundleID;
        SqlDataReader DataReader;
        List<int> LessonBundle_IDs = new List<int>();

        public frmLessonBundle()
        {
            InitializeComponent();
            connectionString =
ConfigurationManager.ConnectionStrings["frmSplash.Properties.Settings.PrivateTuitionCo
nectionString"].ConnectionString;
        }

        private void frmLessonBundle_Load(object sender, EventArgs e)
        {
            Display_BundleDetails();
        }

        // DISPLAY LESSON BUNDLE INFORMATION
        public void Display_BundleDetails()
        {
            using (connection = new SqlConnection(connectionString))
            {
                using (SqlDataAdapter adaptor = new SqlDataAdapter("SELECT * FROM
LessonBundles", connection))
                {
                    DataTable BundleTable = new DataTable();
                    adaptor.Fill(BundleTable);

                    cbBundleSearch.DisplayMember = "Lesson Bundle";
                    cbBundleSearch.ValueMember = "LessonBundleID";
                    cbBundleSearch.DataSource = BundleTable;

                    lbLessonBundleID.DisplayMember = "LessonBundleID";
                    lbLessonBundleID.ValueMember = "LessonBundleID";
                    lbLessonBundleID.DataSource = BundleTable;

                    lbLessonBundleName.DisplayMember = "Lesson Bundle";
                    lbLessonBundleName.ValueMember = "LessonBundleID";
                    lbLessonBundleName.DataSource = BundleTable;
                }
            }
        }
    }
}
```

```
lbBundleCost.DisplayMember = "Bundle Cost";
lbBundleCost.ValueMember = "LessonBundleID";
lbBundleCost.DataSource = BundleTable;

lbDiscount.DisplayMember = "Multiplier (Discount Rate)";
lbDiscount.ValueMember = "LessonBundleID";
lbDiscount.DataSource = BundleTable;

using (SqlCommand Max = new SqlCommand("SELECT MAX(LessonBundleID)
FROM LessonBundles", connection))
{
    connection.Open();
    MaxBundleID = (int)Max.ExecuteScalar();
}

using (SqlCommand Min = new SqlCommand("Select Min(LessonBundleID)
FROM LessonBundles", connection))
{
    MinBundleID = (int)Min.ExecuteScalar();
    connection.Close();
}

}

using (connection = new SqlConnection(connectionString))
{
    connection.Open();
    using (SqlCommand Command = new SqlCommand("SELECT * FROM
LessonBundles", connection))
    {
        DataReader = Command.ExecuteReader();
        while (DataReader.Read())
        {
            // For each record, store the corrisponding peice of
information.
            LessonBundle_IDs.Add(DataReader.GetInt32(0));
        }
    }
    connection.Close();
}

// VIEW PREVIOUS LESSON BUNDLE
private void btnPreviousIBundle_Click(object sender, EventArgs e)
{
    CurrentBundleID = Convert.ToInt32(lbLessonBundleID.SelectedValue);

    if (CurrentBundleID == MinBundleID)
    {
        lbLessonBundleID.SelectedValue = MaxBundleID;
    }
    else
    {
        CurrentBundleID += -1;
        lbLessonBundleID.SelectedValue = CurrentBundleID;
    }
}

// VIEW NEXT LESSON BUNDLE
private void btnNextLessonBundle_Click(object sender, EventArgs e)
{
    CurrentBundleID = Convert.ToInt32(lbLessonBundleID.SelectedValue);
```

```
if (CurrentBundleID == MaxBundleID)
{
    lbLessonBundleID.SelectedValue = MinBundleID;
}
else
{
    CurrentBundleID += 1;
    lbLessonBundleID.SelectedValue = CurrentBundleID;
}
}

// RETURN TO MAIN MENU
private void btnReturn_Click(object sender, EventArgs e)
{
    frmPrivateTuition MainMenu = new frmPrivateTuition();
    MainMenu.Show();
    this.Hide();
}

// ADD NEW RECORD
private void button1_Click(object sender, EventArgs e)
{
    GlobalVariables.PreviousForm = "LessonBundleTable";
    GlobalVariables.Purpose = "Add";

    frmAddField Addscheudle = new frmAddField();
    Addscheudle.Show();
    this.Hide();
}

// REMOVE RECORD
private void btnRemove_Click(object sender, EventArgs e)
{
    // Set associated field.
    GlobalVariables.PreviousForm = "LessonBundleTable";

    // To remove a record, the user needs to login.
    if (GlobalVariables.UserLoggedIn == false)
    {
        // If no user is currently logged in, load login form.
        frmUserLogin Login = new frmUserLogin();
        Login.Show();
        this.Hide();
    }
    else
    {
        // If a user is logged in, load confirmation page.
        GlobalVariables.TableID = Convert.ToInt32(lbLessonBundleID.Text);
        GlobalVariables.FieldNames = lbLessonBundleName.Text;

        frmConfirmation Confirmation = new frmConfirmation();
        Confirmation.Show();
    }
}

// BUTTON SELECT MODIFCATIONS
// Previous bundle
private void btnPreviousIBundle_MouseEnter(object sender, EventArgs e)
{
    btnPreviousIBundle.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
```

```
}

private void btnPreviousIBundle_MouseLeave(object sender, EventArgs e)
{
    btnPreviousIBundle.BackColor = Color.MidnightBlue;
    this.Cursor = Cursors.Arrow;
}

// Next Bundle
private void btnNextLessonBundle_MouseEnter(object sender, EventArgs e)
{
    btnNextLessonBundle.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}

private void btnNextLessonBundle_MouseLeave(object sender, EventArgs e)
{
    btnNextLessonBundle.BackColor = Color.MidnightBlue;
    this.Cursor = Cursors.Arrow;
}

// Add new Bundle
private void btnAddBundle_MouseEnter(object sender, EventArgs e)
{
    btnAddBundle.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}

private void btnAddBundle_MouseLeave(object sender, EventArgs e)
{
    btnAddBundle.BackColor = Color.SeaGreen;
    this.Cursor = Cursors.Arrow;
}

// Remove Bundle
private void btnRemove_MouseEnter(object sender, EventArgs e)
{
    btnRemove.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}

private void btnRemove_MouseLeave(object sender, EventArgs e)
{
    btnRemove.BackColor = Color.Firebrick;
    this.Cursor = Cursors.Arrow;
}

// Return to Main Menu
private void btnReturn_MouseEnter(object sender, EventArgs e)
{
    btnReturn.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}

private void btnReturn_MouseLeave(object sender, EventArgs e)
{
    btnReturn.BackColor = Color.Firebrick;
    this.Cursor = Cursors.Arrow;
}

// First Bundle
private void pbFirstBundle_MouseEnter(object sender, EventArgs e)
```

```
{  
    pbFirstBundle.Image = Properties.Resources.Arrow2_GreyScale1_;  
    this.Cursor = Cursors.Hand;  
}  
  
private void pbFirstBundle_MouseLeave(object sender, EventArgs e)  
{  
    pbFirstBundle.Image = Properties.Resources.Arrow2;  
    this.Cursor = Cursors.Arrow;  
}  
  
// Last Bundle  
private void pbLastBundle_MouseEnter(object sender, EventArgs e)  
{  
    pbLastBundle.Image = Properties.Resources.Arrow2_Inverted_GreyScale1_;  
    this.Cursor = Cursors.Hand;  
}  
  
private void pbLastBundle_MouseLeave(object sender, EventArgs e)  
{  
    pbLastBundle.Image = Properties.Resources.Arrow2_Inverted_;  
    this.Cursor = Cursors.Arrow;  
}  
  
// Update Record  
private void btnUpdate_MouseEnter(object sender, EventArgs e)  
{  
    btnUpdate.BackColor = Color.LightSlateGray;  
    this.Cursor = Cursors.Hand;  
}  
private void btnUpdate_MouseLeave(object sender, EventArgs e)  
{  
    btnUpdate.BackColor = Color.Chocolate;  
    this.Cursor = Cursors.Arrow;  
}  
  
private void btnUpdate_Click(object sender, EventArgs e)  
{  
    GlobalVariables.PreviousForm = "LessonBundleTable";  
    GlobalVariables.Purpose = "Update";  
    GlobalVariables.UpdateID =  
    lbLessonBundleID.GetItemText(lbLessonBundleID.SelectedItem);  
  
    GlobalVariables.Bundle_Name =  
    lbLessonBundleName.GetItemText(lbLessonBundleName.SelectedItem);  
    GlobalVariables.Bundle_Cost =  
    lbBundleCost.GetItemText(lbBundleCost.SelectedItem);  
    GlobalVariables.Bundle_Discount =  
    lbDiscount.GetItemText(lbDiscount.SelectedItem);  
  
    frmAddField UpdateRecord = new frmAddField();  
    UpdateRecord.Show();  
    this.Hide();  
}  
  
// FIRST / LAST RECORD  
private void pbFirstBundle_Click(object sender, EventArgs e)  
{  
    lbLessonBundleID.SelectedValue = LessonBundle_IDs[0];  
}  
private void pbLastBundle_Click(object sender, EventArgs e)  
{
```

```
    lbLessonBundleID.SelectedValue =
LessonBundle_IDs[(LessonBundle_IDs.Count() - 1)];
}
}
```

FrmScheduleTable

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Configuration;
using System.Data.SqlClient;

namespace frmSplash
{
    public partial class frmScheduleTable : Form
    {
        SqlConnection connection;
        string connectionString;
        int CurrentStudentID;
        int MinStudentID;
        int MaxStudentID;
        SqlDataReader DataReader;
        List<int> AllStudentIDs = new List<int>();
        List<string> AllStudent_FirstNames = new List<string>();
        List<string> AllStudent_Surnames = new List<string>();
        List<string> AllStudent_FullNameValues = new List<string>();
        string SearchedStudent;
        char SearchTest;
        string Search_FirstName;
        string Search_Surname;
        List<int> StudentID_FirstNameMatches = new List<int>();
        List<int> StudentID_SurnameMatches = new List<int>();
        int Search_DesiredStudentID;
        bool Firstdisplay;

        List<int> Schedule_ID = new List<int>();
        List<int> Schedule_StudentID = new List<int>();
        List<int> Schedule_TeacherID = new List<int>();
        List<int> Schedule_BundleID = new List<int>();
        List<string> Schedule_Weks = new List<string>();
        List<DateTime> Schedule_StartDate = new List<DateTime>();
        List<string> Schedule_BookedDays = new List<string>();
        List<string> Schedule_BookedTime = new List<string>();
        List<DateTime> Schedule_EndDate = new List<DateTime>();

        public frmScheduleTable()
        {
            InitializeComponent();
            connectionString =
ConfigurationManager.ConnectionStrings["frmSplash.Properties.Settings.PrivateTuitionConnectionString"].ConnectionString;
        }

        private void frmScheduleTable_Load(object sender, EventArgs e)
        {
            btnCalendarReturn.Hide();
            DisplayStudentDetails();
            DisplayTeacherDetails();
            DisplayScheduleDetails();
            DisplayGradeDetails();
```

```
        StoreScheduleInformation();

        if (GlobalVariables.CalendarReturn == true)
        {
            btnCalendarReturn.Show();
        }

        if (GlobalVariables.CalenderStudentID != 0)
        {
            lbStudentID.Text = GlobalVariables.CalenderStudentID.ToString();
            lbScheduledLessons.Text =
GlobalVariables.CalenderScheduledID.ToString();
        }

    }

    // RETURN TO MAIN MENU
    private void btnReturn_Click(object sender, EventArgs e)
    {
        GlobalVariables.CalendarReturn = false;
        frmPrivateTuition PrivateTuition = new frmPrivateTuition();
        PrivateTuition.Show();
        this.Hide();
    }

    // ADD RECORD
    private void button1_Click(object sender, EventArgs e)
    {
        GlobalVariables.PreviousForm = "ScheduledLessonTable";
        GlobalVariables.Purpose = "Add";

        frmAddField Addscheudle = new frmAddField();
        Addscheudle.Show();
        this.Hide();
    }

    // DISPLAY SCHEDULE INFORMATION
    public void DisplayScheduleDetails()
    {
        using (connection = new SqlConnection(connectionString))
        using (SqlCommand command = new SqlCommand("SELECT * FROM
Scheduled_Lessons WHERE StudentID = @Value", connection))
        {
            command.Parameters.Add(new SqlParameter("@value", lbStudentID.Text));

            using (SqlDataAdapter Adaptor = new SqlDataAdapter())
            {
                DataTable ScheduledDetails = new DataTable();
                Adaptor.SelectCommand = command;
                Adaptor.Fill(ScheduledDetails);

                lbTeacherID.DisplayMember = "TeacherID";
                lbTeacherID.ValueMember = "ScheduleID";
                lbTeacherID.DataSource = ScheduledDetails;

                lbScheduledLessons.DisplayMember = "ScheduleID";
                lbScheduledLessons.ValueMember = "ScheduleID";
                lbScheduledLessons.DataSource = ScheduledDetails;

                dataGridView1.DataSource = ScheduledDetails;
            }
        }
    }
}
```

```
}

// DISPLAY STUDENT INFORMATION
public void DisplayStudentDetails()
{
    using (connection = new SqlConnection(connectionString))
        using (SqlDataAdapter adaptor = new SqlDataAdapter("SELECT * FROM Students
ORDER BY First_Name", connection))
    {
        DataTable StudentTable = new DataTable();
        adaptor.Fill(StudentTable);

        lbStudentID.DisplayMember = "StudentID";
        lbStudentID.ValueMember = "StudentID";
        lbStudentID.DataSource = StudentTable;

        lbFirstName.DisplayMember = "First_Name";
        lbFirstName.ValueMember = "StudentID";
        lbFirstName.DataSource = StudentTable;

        lbSurname.DisplayMember = "Surname";
        lbSurname.ValueMember = "StudentID";
        lbSurname.DataSource = StudentTable;

        lbContactNumber.DisplayMember = "Contact_Number";
        lbContactNumber.ValueMember = "StudentID";
        lbContactNumber.DataSource = StudentTable;

        lbGradeID.DisplayMember = "GradeID";
        lbGradeID.ValueMember = "StudentID";
        lbGradeID.DataSource = StudentTable;

        using (connection = new SqlConnection(connectionString))
        {
            using (SqlCommand Max = new SqlCommand("SELECT MAX(StudentID) FROM
Students", connection))
            {
                connection.Open();
                MaxStudentID = (int)Max.ExecuteScalar();
                connection.Close();
            }

            using (connection = new SqlConnection(connectionString))
            {
                using (SqlCommand Min = new SqlCommand("Select Min(StudentID)
FROM Students", connection))
                {
                    connection.Open();
                    MinStudentID = (int)Min.ExecuteScalar();
                    connection.Close();
                }

                StoreAllStudentIDs();
                Fill_StudentSearchBox();
            }
        }
    }
}

public void StoreScheduleInformation()
{
```

```
        using (connection = new SqlConnection(connectionString))
    {
        using (SqlCommand cmd = new SqlCommand("Select * FROM
Scheduled_Lessons", connection))
        {
            connection.Open();
            DataReader = cmd.ExecuteReader();

            while (DataReader.Read())
            {
                Schedule_ID.Add(DataReader.GetInt32(0));
                Schedule_StudentID.Add(DataReader.GetInt32(1));
                Schedule_TeacherID.Add(DataReader.GetInt32(2));
                Schedule_BundleID.Add(DataReader.GetInt32(3));
                Schedule_Weeks.Add(DataReader.GetString(4));
                Schedule_StartDate.Add(DataReader.GetDateTime(5));
                Schedule_BookedDays.Add(DataReader.GetString(6));
                Schedule_BookedTime.Add(DataReader.GetString(7));
                Schedule_EndDate.Add(DataReader.GetDateTime(8));
            }

            connection.Close();
        }
    }
}

// DISPLAY GRADE INFORMATION
public void DisplayGradeDetails()
{
    using (connection = new SqlConnection(connectionString))
    using (SqlCommand Gradecommand = new SqlCommand("SELECT * FROM Grade
WHERE GradeID = @ID", connection))
    {
        Gradecommand.Parameters.Add(new SqlParameter("@ID", lbGradeID.Text));

        using (SqlDataAdapter GradeAdaptor = new SqlDataAdapter())
        {
            connection.Open();

            DataTable Grades = new DataTable();
            GradeAdaptor.SelectCommand = Gradecommand;
            GradeAdaptor.Fill(Grades);

            lbGrade.DisplayMember = "GradeLevel";
            lbGrade.ValueMember = "GradeID";
            lbGrade.DataSource = Grades;
        }
    }
}

// DISPLAY TEACHER INFORMATION
public void DisplayTeacherDetails()
{
    using (connection = new SqlConnection(connectionString))
    using (SqlCommand command = new SqlCommand("SELECT * FROM Teachers WHERE
TeacherID = @ID", connection))
    {
        command.Parameters.Add(new SqlParameter("@ID", lbTeacherID.Text));

        using (SqlDataAdapter Adaptor = new SqlDataAdapter())
        {
```

```
        DataTable TeacherTable = new DataTable();
        Adaptor.SelectCommand = command;
        Adaptor.Fill(TeacherTable);

        lbTeacherFirst_Name.DisplayMember = "First_Name";
        lbTeacherFirst_Name.ValueMember = "TeacherID";
        lbTeacherFirst_Name.DataSource = TeacherTable;

        lbTeacherSurname.DisplayMember = "Surname";
        lbTeacherSurname.ValueMember = "TeacherID";
        lbTeacherSurname.DataSource = TeacherTable;

        lbSpecialisation.DisplayMember = "Specialisation";
        lbSpecialisation.ValueMember = "TeacherID";
        lbSpecialisation.DataSource = TeacherTable;

        lbRoomID.DisplayMember = "RoomID";
        lbRoomID.ValueMember = "TeacherID";
        lbRoomID.DataSource = TeacherTable;
    }
}

private void lbStudentID_SelectedIndexChanged(object sender, EventArgs e)
{
    DisplayScheduleDetails();
    DisplayTeacherDetails();
    DisplayGradeDetails();
    dataGridView1.AutoSizeColumnsMode =
DataGridViewAutoSizeColumnsMode.DisplayedCells;
}

// VIEW NEXT RECORD
private void btnNextStudent_Click(object sender, EventArgs e)
{
    CurrentStudentID = Convert.ToInt32(lbStudentID.Text);

    if (CurrentStudentID == MaxStudentID)
    {
        lbStudentID.SelectedValue = MinStudentID;
    }
    else
    {
        CurrentStudentID += 1;
        lbStudentID.SelectedValue = CurrentStudentID;
    }
}

// VIEW PREVIOUS RECORD
private void btnPreviousStudent_Click(object sender, EventArgs e)
{
    CurrentStudentID = Convert.ToInt32(lbStudentID.Text);

    if (CurrentStudentID == MinStudentID)
    {
        lbStudentID.SelectedValue = MaxStudentID;
    }
    else
    {
        CurrentStudentID += -1;
    }
}
```

```
        lbStudentID.SelectedValue = CurrentStudentID;
    }
}

// RETURN TO CALENDAR
private void button2_Click(object sender, EventArgs e)
{
    GlobalVariables.scheduleErrorMessage = false;
    frmClassSchedule ClassSchedule = new frmClassSchedule();
    ClassSchedule.Show();
    this.Hide();
}

// REMOVE RECORD
private void btnRemove_Click(object sender, EventArgs e)
{
    // Set associated field.
    GlobalVariables.PreviousForm = "ScheduledLessonTable";

    // To remove a record, the user needs to login.
    if (GlobalVariables.UserLoggedIn == false)
    {
        // If no user is currently logged in, load login form.
        frmUserLogin Login = new frmUserLogin();
        Login.Show();
        this.Hide();
    }
    else
    {
        if (lbScheduledLessons.Text != null &&
!string.IsNullOrWhiteSpace(lbScheduledLessons.Text))
        {
            // If a user is logged in, load confirmation page.
            GlobalVariables.TableID =
Convert.ToInt32(lbScheduledLessons.Text);
            GlobalVariables.FieldNames = lbFirstName.Text + " " +
lbSurname.Text;

            frmConfirmation Confirmation = new frmConfirmation();
            Confirmation.Show();
        }
        else
        {
            MessageBox.Show("Please select a scheduled Lesson Reocrd from the
menu below. If none are present, then this student has no scheduled lessons.");
        }
    }
}

// STORE STUDENT INFORMATION
public void StoreAllStudentIDs()
{
    // This fuunction will be used to store various peices of information from
the schedule Table. This will be used to complete the schedule
    using (connection = new SqlConnection(connectionString))
    {
        connection.Open();
        using (SqlCommand Command = new SqlCommand("SELECT * FROM Students",
connection))
        {
            DataReader = Command.ExecuteReader();
            while (DataReader.Read())

```

```
        {
            // For each record, store the corrisponding peice of
            information.
            AllStudentIDs.Add(DataReader.GetInt32(0));
            AllStudent_FirstNames.Add(DataReader.GetString(1));
            AllStudent_Surnames.Add(DataReader.GetString(3));
        }
    }
    connection.Close();
}

// FILL SEARCH BOX
public void Fill_StudentSearchBox()
{
    for (int x = 0; x < AllStudentIDs.Count(); x++)
    {
        AllStudent_FullNameValues.Add(AllStudent_FirstNames[x] + " " +
AllStudent_Surnames[x]);
    }

    AllStudent_FullNameValues.Sort();
    cbStudentID.DataSource = AllStudent_FullNameValues;
}

// STUDENT SEARCH FUNCTION
private void cbStudentID_SelectedIndexChanged_1(object sender, EventArgs e)
{
    StudentID_FirstNameMatches.Clear();
    StudentID_SurnameMatches.Clear();

    // identifies and stores selected value.
    SearchedStudent = cbStudentID.SelectedValue.ToString();

    for (int Y = 0; Y < SearchedStudent.Length; Y++)
    {
        // Then take each charater present within the string, and seperately
        look for a 'Space'.
        SearchTest = Convert.ToChar(SearchedStudent.Substring(Y, 1));
        if (SearchTest == ' ')
        {
            // If located, seperate first name from surname.
            Search_FirstName = SearchedStudent.Substring(0, (Y));
            Search_Surname = SearchedStudent.Substring((Y + 1),
(SearchedStudent.Length - (Search_FirstName.Length + 1)));
            break;
        }
    }

    // Then proceed to take the first name, and compare it against all first
    names in the system.
    for (int x = 0; x < AllStudent_FirstNames.Count(); x++)
    {
        // If their is a match, record the associated student ID.
        if (AllStudent_FirstNames[x] == Search_FirstName)
        {
            StudentID_FirstNameMatches.Add(AllStudentIDs[x]);
        }
    }
}
```

```
// Then compare the collected surname against the student table
information.
for (int z = 0; z < AllStudent_Surnames.Count(); z++)
{
    // if there is a match, then once again categorise the associated
student ID.
    if (AllStudent_Surnames[z] == Search_Surname)
    {
        StudentID_SurnameMatches.Add(AllStudentIDs[z]);
    }
}

// This block of code will be used to handle the scenario in which
multiple students possess the same first, or surnames.
// I will compare all collected student IDs, for a match should only occur
should the desired student be located.
for (int x = 0; x < StudentID_FirstNameMatches.Count(); x++)
{
    for (int y = 0; y < StudentID_SurnameMatches.Count(); y++)
    {
        if (StudentID_FirstNameMatches[x] == StudentID_SurnameMatches[y])
        {
            // Depending on whether the system contains students with
matching firstnames or surnames, list will contain different values.
            // For this reason, the system will try both possibilities,
should one fail to work.
            try
            {
                Search_DesiredStudentID = StudentID_FirstNameMatches[x];
            }
            catch (Exception SearchFail)
            {
                Search_DesiredStudentID = StudentID_SurnameMatches[y];
            }
        }
    }
}

// Then finally, update the form as to display the information for the
selected student.
lbStudentID.SelectedValue = Search_DesiredStudentID;

if (Firstdisplay == false)
{
    lbStudentID.SelectedValue = AllStudentIDs[0];
    Firstdisplay = true;
}
}

// BUTTON SELECT MODIFICATIONS
// Previous Record
private void btnPreviousStudent_MouseEnter(object sender, EventArgs e)
{
    btnPreviousStudent.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}

private void btnPreviousStudent_MouseLeave(object sender, EventArgs e)
{
    btnPreviousStudent.BackColor = Color.MidnightBlue;
    this.Cursor = Cursors.Arrow;
```

```
}

// Next Record
private void btnNextStudent_MouseEnter(object sender, EventArgs e)
{
    btnNextStudent.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}

private void btnNextStudent_MouseLeave(object sender, EventArgs e)
{
    btnNextStudent.BackColor = Color.MidnightBlue;
    this.Cursor = Cursors.Arrow;
}

// Add New Record
private void button1_MouseEnter(object sender, EventArgs e)
{
    btnAddRecord.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}

private void button1_MouseLeave(object sender, EventArgs e)
{
    btnAddRecord.BackColor = Color.SeaGreen;
    this.Cursor = Cursors.Arrow;
}

// Remove Record
private void btnRemove_MouseEnter(object sender, EventArgs e)
{
    btnRemove.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}

private void btnRemove_MouseLeave(object sender, EventArgs e)
{
    btnRemove.BackColor = Color.Firebrick;
    this.Cursor = Cursors.Arrow;
}

// Calendar Return
private void btnCalendarReturn_MouseEnter(object sender, EventArgs e)
{
    btnCalendarReturn.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}

private void btnCalendarReturn_MouseLeave(object sender, EventArgs e)
{
    btnCalendarReturn.BackColor = Color.Chocolate;
    this.Cursor = Cursors.Arrow;
}

// Main Menu Return
private void btnReturn_MouseEnter(object sender, EventArgs e)
{
    btnReturn.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}

private void btnReturn_MouseLeave(object sender, EventArgs e)
```

```
{  
    btnReturn.BackColor = Color.Firebrick;  
    this.Cursor = Cursors.Arrow;  
}  
  
// First Record  
private void pbFirstRecord_MouseEnter(object sender, EventArgs e)  
{  
    pbFirstRecord.Image = Properties.Resources.Arrow2_GreyScale1_;  
    this.Cursor = Cursors.Hand;  
}  
  
private void pbFirstRecord_MouseLeave(object sender, EventArgs e)  
{  
    pbFirstRecord.Image = Properties.Resources.Arrow2;  
    this.Cursor = Cursors.Arrow;  
}  
  
// Last Record  
private void pbLastRecord_MouseEnter(object sender, EventArgs e)  
{  
    pbLastRecord.Image = Properties.Resources.Arrow2_Inverted_GreyScale1_;  
    this.Cursor = Cursors.Hand;  
}  
  
private void pbLastRecord_MouseLeave(object sender, EventArgs e)  
{  
    pbLastRecord.Image = Properties.Resources.Arrow2_Inverted_;  
    this.Cursor = Cursors.Arrow;  
}  
  
// Update Record  
private void btnUpdate_MouseEnter(object sender, EventArgs e)  
{  
    btnUpdate.BackColor = Color.LightSlateGray;  
    this.Cursor = Cursors.Hand;  
}  
private void btnUpdate_MouseLeave(object sender, EventArgs e)  
{  
    btnUpdate.BackColor = Color.Chocolate;  
    this.Cursor = Cursors.Arrow;  
}  
  
private void btnUpdate_Click(object sender, EventArgs e)  
{  
    if (lbScheduledLessons.Text != null &&  
!string.IsNullOrWhiteSpace(lbScheduledLessons.Text))  
    {  
        GlobalVariables.PreviousForm = "ScheduledLessonTable";  
        GlobalVariables.Purpose = "Update";  
        GlobalVariables.UpdateID =  
lbScheduledLessons.GetItemText(lbScheduledLessons.SelectedItem);  
  
        for (int x = 0; x < Schedule_ID.Count(); x++)  
        {  
            if  
(Convert.ToInt32(lbScheduledLessons.GetItemText(lbScheduledLessons.SelectedItem)) ==  
Schedule_ID[x])  
            {  
                GlobalVariables.Schedule_StudentID =  
Convert.ToString(Schedule_StudentID[x]);  
            }  
        }  
    }  
}
```

```
        GlobalVariables.Schedule_TeacherID =
Convert.ToString(Schedule_TeacherID[x]);
        GlobalVariables.Schedule_PurchaseID =
Convert.ToString(Schedule_BundleID[x]);
        GlobalVariables.Schedule_Weeks = Schedule_Weeks[x];
        GlobalVariables.Schedule_StartDate =
Schedule_StartDate[x].ToString();
        GlobalVariables.Schedule_BookedDays = Schedule_BookedDays[x];
        GlobalVariables.Schedule_BookedTime = Schedule_BookedTime[x];
        GlobalVariables.Schedule_EndDate =
Schedule_EndDate[x].ToString();
        break;
    }
}

frmAddField UpdateRecord = new frmAddField();
UpdateRecord.Show();
this.Hide();
}
else
{
    MessageBox.Show("Please select a Scheduled Lesson Record. (Select from
Record Display)");
}
}

// FIRST / LAST RECORD
private void pbFirstRecord_Click(object sender, EventArgs e)
{
    lbStudentID.SelectedValue = AllStudentIDs[0];
}
private void pbLastRecord_Click(object sender, EventArgs e)
{
    lbStudentID.SelectedValue = AllStudentIDs[(AllStudentIDs.Count() - 1)];
}
}
```

PurchasedLessons

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Configuration;
using System.Data.SqlClient;

namespace frmSplash
{
    public partial class PurchasedLessonBundles : Form
    {
        SqlConnection connection;
        string connectionString;
        int CurrentStudentID;
        int MinStudentID;
        int MaxStudentID;
        SqlDataReader DataReader;
        List<int> AllScheduleLessonBundleIDs = new List<int>();
        List<int> AllLessonBundleIDs = new List<int>();
        List<int> AllLessonCosts = new List<int>();
        List<double> ALLLessonDiscounts = new List<double>();
        List<int> AllPurchase_StudentID = new List<int>();
        List<int> AllStudnetGradeID = new List<int>();
        List<int> AllGradeFee = new List<int>();

        List<int> AllStudentIDs = new List<int>();
        List<string> AllStudent_FirstNames = new List<string>();
        List<string> AllStudent_Surnames = new List<string>();
        List<string> AllStudent_FullNameValues = new List<string>();
        string SearchedStudent;
        char SearchTest;
        string Search_FirstName;
        string Search_Surname;
        List<int> StudentID_FirstNameMatches = new List<int>();
        List<int> StudentID_SurnameMatches = new List<int>();
        int Search_DesiredStudentID;
        bool Firstdisplay;

        List<int> PurchasedLessons_StudentID = new List<int>();
        List<int> PurchasedLessons_LessonBundleID = new List<int>();
        List<DateTime> PurchasedLessons_PaymentDate = new List<DateTime>();
        List<string> PurchasedLessons_Method = new List<string>();
        List<bool> PurchasedLessons_Recieved = new List<bool>();
        List<string> PurchasedLessons_RecievedDate = new List<string>();
        List<decimal> PurchasedLessons_BundleCost = new List<decimal>();

        public PurchasedLessonBundles()
        {
            InitializeComponent();
            connectionString =
ConfigurationManager.ConnectionStrings["frmSplash.Properties.Settings.PrivateTuitionCo
nnectionString"].ConnectionString;
        }

        private void PurchasedLessonBundles_Load(object sender, EventArgs e)
```

```
{  
    StoreAllStudentIDs();  
    Fill_StudentSearchBox();  
    StorePurchasedInformation();  
    DisplayStudentDetails();  
    DisplayPurchaseDetails();  
    DisplayBundleDetails();  
    DisplayGradeDetails();  
    DataGridPurchasedLessons.AutoSizeColumnsMode =  
    DataGridViewAutoSizeColumnsMode.DisplayedCells;  
}  
  
public void DisplayPurchaseDetails()  
{  
    using (connection = new SqlConnection(connectionString))  
        using (SqlCommand command = new SqlCommand("SELECT * FROM LessonsPurchased  
WHERE StudentID = @Value", connection))  
    {  
        command.Parameters.Add(new SqlParameter("@value", lbStudentID.Text));  
  
        using (SqlDataAdapter Adaptor = new SqlDataAdapter())  
        {  
            DataTable PurchaseDetails = new DataTable();  
            Adaptor.SelectCommand = command;  
            Adaptor.Fill(PurchaseDetails);  
  
            lbBundleID.DisplayMember = "LessonBundleID";  
            lbBundleID.ValueMember = "LessonsPurchasedID";  
            lbBundleID.DataSource = PurchaseDetails;  
  
            lbPurchaseID.DisplayMember = "LessonsPurchasedID";  
            lbPurchaseID.ValueMember = "LessonsPurchasedID";  
            lbPurchaseID.DataSource = PurchaseDetails;  
  
            DataGridPurchasedLessons.DataSource = PurchaseDetails;  
        }  
    }  
}  
  
public void StorePurchasedInformation()  
{  
    using (connection = new SqlConnection(connectionString))  
    {  
        using (SqlCommand cmd = new SqlCommand("Select * FROM  
LessonsPurchased", connection))  
        {  
            connection.Open();  
            DataReader = cmd.ExecuteReader();  
  
            while (DataReader.Read())  
            {  
                PurchasedLessons_StudentID.Add(DataReader.GetInt32(1));  
                PurchasedLessons_LessonBundleID.Add(DataReader.GetInt32(2));  
                PurchasedLessons_PaymentDate.Add(DataReader.GetDateTime(3));  
                PurchasedLessons_Method.Add(DataReader.GetString(4));  
                PurchasedLessons_Recieved.Add(DataReader.GetBoolean(5));  
                PurchsedLessons_RecievedDate.Add(DataReader.GetString(6));  
                PurchasedLessons_BundleCost.Add(DataReader.GetDecimal(7));  
            }  
  
            connection.Close();  
        }  
    }  
}
```

```
    }

    }

    public void DisplayStudentDetails()
    {
        using (connection = new SqlConnection(connectionString))
        using (SqlDataAdapter adaptor = new SqlDataAdapter("SELECT * FROM
Students", connection))
        {
            DataTable StudentTable = new DataTable();
            adaptor.Fill(StudentTable);

            lbStudentID.DisplayMember = "StudentID";
            lbStudentID.ValueMember = "StudentID";
            lbStudentID.DataSource = StudentTable;

            lbFirstName.DisplayMember = "First_Name";
            lbFirstName.ValueMember = "StudentID";
            lbFirstName.DataSource = StudentTable;

            lbSurname.DisplayMember = "Surname";
            lbSurname.ValueMember = "StudentID";
            lbSurname.DataSource = StudentTable;

            lbContactNumber.DisplayMember = "Contact_Number";
            lbContactNumber.ValueMember = "StudentID";
            lbContactNumber.DataSource = StudentTable;

            lbGradeID.DisplayMember = "GradeID";
            lbGradeID.ValueMember = "StudentID";
            lbGradeID.DataSource = StudentTable;

            using (connection = new SqlConnection(connectionString))
            using (SqlCommand Max = new SqlCommand("SELECT MAX(StudentID) FROM
Students", connection))
            {
                connection.Open();
                MaxStudentID = (int)Max.ExecuteScalar();
                connection.Close();
            }

            using (connection = new SqlConnection(connectionString))
            using (SqlCommand Min = new SqlCommand("Select Min(StudentID) FROM
Students", connection))
            {
                connection.Open();
                MinStudentID = (int)Min.ExecuteScalar();
                connection.Close();
            }
        }
    }

    public void DisplayGradeDetails()
    {
        using (connection = new SqlConnection(connectionString))
        using (SqlCommand GradeCommand = new SqlCommand("SELECT * FROM Grade
WHERE GradeID = @ID", connection))
        {
```

```
Gradecommand.Parameters.Add(new SqlParameter("@ID", lbGradeID.Text));

using (SqlDataAdapter GradeAdaptor = new SqlDataAdapter())
{
    connection.Open();

    DataTable Grades = new DataTable();
    GradeAdaptor.SelectCommand = Gradecommand;
    GradeAdaptor.Fill(Grades);

    lbGrade.DisplayMember = "GradeLevel";
    lbGrade.ValueMember = "GradeID";
    lbGrade.DataSource = Grades;
}
}

public void DisplayBundleDetails()
{

    using (connection = new SqlConnection(connectionString))
        using (SqlCommand command = new SqlCommand("SELECT * FROM LessonBundles
WHERE LessonBundleID = @Value", connection))
    {
        command.Parameters.Add(new SqlParameter("@value", lbBundleID.Text));
        using (SqlDataAdapter Adaptor = new SqlDataAdapter())
        {
            DataTable BundleDetails = new DataTable();
            Adaptor.SelectCommand = command;
            Adaptor.Fill(BundleDetails);

            lbLessonBundle.DisplayMember = "Lesson Bundle";
            lbLessonBundle.ValueMember = "LessonsPurchasedID";
            lbLessonBundle.DataSource = BundleDetails;

            lbBundleCost.DisplayMember = "Bundle Cost";
            lbBundleCost.ValueMember = "LessonsPurchasedID";
            lbBundleCost.DataSource = BundleDetails;

            lbCostMultiplier.DisplayMember = "Multiplier (Discount Rate)";
            lbCostMultiplier.ValueMember = "LessonsPurchasedID";
            lbCostMultiplier.DataSource = BundleDetails;
        }
    }
}

private void lbStudentID_SelectedIndexChanged(object sender, EventArgs e)
{
    DisplayPurchaseDetails();
    DisplayBundleDetails();
    DisplayGradeDetails();
}

private void btnPreviousStudent_Click(object sender, EventArgs e)
{
    CurrentStudentID = Convert.ToInt32(lbStudentID.Text);

    if (CurrentStudentID == MinStudentID)
    {
```

```
        lbStudentID.SelectedValue = MaxStudentID;
    }
    else
    {
        CurrentStudentID += -1;
        lbStudentID.SelectedValue = CurrentStudentID;
    }
}

private void btnNextStudent_Click(object sender, EventArgs e)
{
    CurrentStudentID = Convert.ToInt32(lbStudentID.Text);

    if (CurrentStudentID == MaxStudentID)
    {
        lbStudentID.SelectedValue = MinStudentID;
    }
    else
    {
        CurrentStudentID += 1;
        lbStudentID.SelectedValue = CurrentStudentID;
    }
}

private void btnAddPurchase_Click(object sender, EventArgs e)
{
    GlobalVariables.PreviousForm = "PurchasedLessonBundleTable";

    frmAddField AddPurchase = new frmAddField();
    AddPurchase.Show();
    this.Hide();
}

private void btnReturn_Click(object sender, EventArgs e)
{
    frmPrivateTuition Menu = new frmPrivateTuition();
    Menu.Show();
    this.Hide();
}

private void lbPurchaseID_SelectedIndexChanged(object sender, EventArgs e)
{
    DisplayBundleDetails();
}

private void btnRemove_Click(object sender, EventArgs e)
{
    // Set associated field.
    GlobalVariables.PreviousForm = "PurchasedLessonsTable";

    // To remove a record, the user needs to login.
    if (GlobalVariables.UserLoggedIn == false)
    {
        // If no user is currently logged in, load login form.
        frmUserLogin Login = new frmUserLogin();
        Login.Show();
        this.Hide();
    }
    else
    {
        // If a user is logged in, load confirmation page.
```

```
        if (lbPurchaseID != null &&
!string.IsNullOrWhiteSpace(lbPurchaseID.Text))
{
    GlobalVariables.TableID = Convert.ToInt32(lbPurchaseID.Text);
    GlobalVariables.FieldNames = lbFirstName.Text + " " +
lbSurname.Text;

    frmConfirmation Confirmation = new frmConfirmation();
    Confirmation.Show();
}
else
{
    MessageBox.Show("Please select a Lesson from the menu below. If
none are present, then this student has not purchased any lessons.");
}

}

private void btnCalculate_Click(object sender, EventArgs e)
{

}

// STORE STUDENT INFORMATION
public void StoreAllStudentIDs()
{
    // This function will be used to store various pieces of information from
the schedule Table. This will be used to complete the schedule
    using (connection = new SqlConnection(connectionString))
    {
        connection.Open();
        using (SqlCommand Command = new SqlCommand("SELECT * FROM Students",
connection))
        {
            DataReader = Command.ExecuteReader();
            while (DataReader.Read())
            {
                // For each record, store the corresponding piece of
information.
                AllStudentIDs.Add(DataReader.GetInt32(0));
                AllStudent_FirstNames.Add(DataReader.GetString(1));
                AllStudent_Surnames.Add(DataReader.GetString(3));
            }
        }
        connection.Close();
    }
}

// FILL SEARCH BOX
public void Fill_StudentSearchBox()
{
    for (int x = 0; x < AllStudentIDs.Count(); x++)
    {
        AllStudent_FullNameValues.Add(AllStudent_FirstNames[x] + " " +
AllStudent_Surnames[x]);
    }

    AllStudent_FullNameValues.Sort();
    cbStudentID.DataSource = AllStudent_FullNameValues;
}
```

```
private void cbStudentID_SelectedIndexChanged(object sender, EventArgs e)
{
    StudentID_FirstNameMatches.Clear();
    StudentID_SurnameMatches.Clear();

    // identifies and stores selected value.
    SearchedStudent = cbStudentID.SelectedValue.ToString();

    for (int Y = 0; Y < SearchedStudent.Length; Y++)
    {
        // Then take each character present within the string, and separately
        look for a 'Space'.
        SearchTest = Convert.ToChar(SearchedStudent.Substring(Y, 1));
        if (SearchTest == ' ')
        {
            // If located, separate first name from surname.
            Search_FirstName = SearchedStudent.Substring(0, (Y));
            Search_Surname = SearchedStudent.Substring((Y + 1),
                (SearchedStudent.Length - (Search_FirstName.Length + 1)));
            break;
        }
    }

    // Then proceed to take the first name, and compare it against all first
    names in the system.
    for (int x = 0; x < AllStudent_FirstNames.Count(); x++)
    {
        // If there is a match, record the associated student ID.
        if (AllStudent_FirstNames[x] == Search_FirstName)
        {
            StudentID_FirstNameMatches.Add(AllStudentIDs[x]);
        }
    }

    // Then compare the collected surname against the student table
    information.
    for (int z = 0; z < AllStudent_Surnames.Count(); z++)
    {
        // if there is a match, then once again categorise the associated
        student ID.
        if (AllStudent_Surnames[z] == Search_Surname)
        {
            StudentID_SurnameMatches.Add(AllStudentIDs[z]);
        }
    }

    // This block of code will be used to handle the scenario in which
    multiple students possess the same first, or surnames.
    // I will compare all collected student IDs, for a match should only occur
    should the desired student be located.
    for (int x = 0; x < StudentID_FirstNameMatches.Count(); x++)
    {
        for (int y = 0; y < StudentID_SurnameMatches.Count(); y++)
        {
            if (StudentID_FirstNameMatches[x] == StudentID_SurnameMatches[y])
            {
                // Depending on whether the system contains students with
                matching firstnames or surnames, list will contain different values.
                // For this reason, the system will try both possibilities,
                should one fail to work.
            }
        }
    }
}
```

```
        try
        {
            Search_DesiredStudentID = StudentID_FirstNameMatches[x];
        }
        catch (Exception SearchFail)
        {
            Search_DesiredStudentID = StudentID_SurnameMatches[y];
        }
    }

}

// Then fianlly, update the form as  to display the information for the
selected student.
lbStudentID.SelectedValue = Search_DesiredStudentID;

if (Firstdisplay == false)
{
    lbStudentID.SelectedValue = AllStudentIDs[0];
    Firstdisplay = true;
}

// BUTTON SELECT MODIFCATIONS
// Previous Record
private void btnPreviousStudent_MouseEnter(object sender, EventArgs e)
{
    btnPreviousStudent.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}
private void btnPreviousStudent_MouseLeave(object sender, EventArgs e)
{
    btnPreviousStudent.BackColor = Color.MidnightBlue;
    this.Cursor = Cursors.Arrow;
}

// Next Record
private void btnNextStudent_MouseEnter(object sender, EventArgs e)
{
    btnNextStudent.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}
private void btnNextStudent_MouseLeave(object sender, EventArgs e)
{
    btnNextStudent.BackColor = Color.MidnightBlue;
    this.Cursor = Cursors.Arrow;
}

// Add Record
private void btnAddPurchase_MouseEnter(object sender, EventArgs e)
{
    btnAddPurchase.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}
private void btnAddPurchase_MouseLeave(object sender, EventArgs e)
{
    btnAddPurchase.BackColor = Color.SeaGreen;
    this.Cursor = Cursors.Arrow;
}

// Remove Record
```

```
private void btnRemove_MouseEnter(object sender, EventArgs e)
{
    btnRemove.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}
private void btnRemove_MouseLeave(object sender, EventArgs e)
{
    btnRemove.BackColor = Color.Firebrick;
    this.Cursor = Cursors.Arrow;
}

// Return To Main Menu
private void btnReturn_MouseEnter(object sender, EventArgs e)
{
    btnReturn.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}
private void btnReturn_MouseLeave(object sender, EventArgs e)
{
    btnReturn.BackColor = Color.Firebrick;
    this.Cursor = Cursors.Arrow;
}

// First Record
private void pbFirstRecord_MouseEnter(object sender, EventArgs e)
{
    pbFirstRecord.Image = Properties.Resources.Arrow2_GreyScale1_;
    this.Cursor = Cursors.Hand;
}
private void pbFirstRecord_MouseLeave(object sender, EventArgs e)
{
    pbFirstRecord.Image = Properties.Resources.Arrow2;
    this.Cursor = Cursors.Arrow;
}

// Last Record
private void pbLastRecord_MouseEnter(object sender, EventArgs e)
{
    pbLastRecord.Image = Properties.Resources.Arrow2_Inverted__GreyScale1_;
    this.Cursor = Cursors.Hand;
}
private void pbLastRecord_MouseLeave(object sender, EventArgs e)
{
    pbLastRecord.Image = Properties.Resources.Arrow2_Inverted_;
    this.Cursor = Cursors.Arrow;
}

private void btnUpdate_Click(object sender, EventArgs e)
{
    if (lbPurchaseID != null && !string.IsNullOrWhiteSpace(lbPurchaseID.Text))
    {
        for (int x = 0; x < PurchasedLessons_StudentID.Count(); x++)
        {
            if (PurchasedLessons_StudentID[x].ToString() == lbStudentID.Text)
            {
                GlobalVariables.PreviousForm = "PurchasedLessonBundleTable";
                GlobalVariables.Purpose = "Update";
                GlobalVariables.UpdateID =
                    lbPurchaseID.GetItemText(lbPurchaseID.SelectedItem);
                GlobalVariables.Purchase_StudentID =
                    PurchasedLessons_StudentID[x].ToString();
            }
        }
    }
}
```

```
        GlobalVariables.Purchase_BundleID =
PurchasedLessons_LessonBundleID[x].ToString();
        GlobalVariables.Purchase_Date =
PurchasedLessons_PaymentDate[x].ToShortDateString();
        GlobalVariables.Purchase_Method = PurchasedLessons_Method[x];
        GlobalVariables.Purchase_Recieved =
PurchasedLessons_Recieved[x].ToString();
        GlobalVariables.Purchase_ReceivedDate =
PurchasedLessons_RecievedDate[x];
        GlobalVariables.Purchase_BundleCosts =
PurchasedLessons_BundleCost[x].ToString();

        frmAddField Add = new frmAddField();
        Add.Show();
        this.Hide();
    }
}

}
else
{
    MessageBox.Show("Please select a Purchased Lesson Record.");
}
}

// Update Record
private void btnUpdate_MouseEnter(object sender, EventArgs e)
{
    btnUpdate.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}
private void btnUpdate_MouseLeave(object sender, EventArgs e)
{
    btnUpdate.BackColor = Color.Chocolate;
    this.Cursor = Cursors.Arrow;
}

private void pbFirstRecord_Click(object sender, EventArgs e)
{
    lbStudentID.SelectedValue = AllStudentIDs[0];
}
private void pbLastRecord_Click(object sender, EventArgs e)
{
    lbStudentID.SelectedValue = AllStudentIDs[(AllStudentIDs.Count() - 1)];
}
}
```

FrmAddField

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Configuration;
using System.Data.SqlClient;

namespace frmSplash
{
    public partial class frmAddField : Form
    {
        List<int> AllScheduledLessonsIDs = new List<int>();
        List<int> AllScheduledLessonsStudentIDs = new List<int>();
        List<int> AllScheduledLessonsTeacherIDs = new List<int>();
        List<int> AllScheduledLessonsBundleIDs = new List<int>();
        List<string> AllScheduledBookedDays = new List<string>();
        List<DateTime> AllScheduledLessonsStartDates = new List<DateTime>();
        List<DateTime> AllScheduledLessonsEndDates = new List<DateTime>();
        List<string> AllStartWeekDays = new List<string>();
        List<int> AllLessonDatesScheduleIDs = new List<int>();

        List<int> PurchasedLesson_IDs = new List<int>();
        List<int> PurchasedLessons_StudentID = new List<int>();
        List<int> PurchasedLessons_LessonBundleID = new List<int>();
        List<int> Student_ID = new List<int>();
        List<int> Student_GradeID = new List<int>();
        List<int> LessonBundle_ID = new List<int>();
        List<int> LessonBundle_BundleCost = new List<int>();
        List<double> LessonBundle_Multiplier = new List<double>();
        List<int> Grade_ID = new List<int>();
        List<int> Grade_Cost = new List<int>();
        List<int> LessonDates_Archive_ID = new List<int>();

        int Desired_PurchasedLesson_StudentID;
        int Desired_PurchasedLesson_ID;
        int Desired_PurchasedLessons_LessonBundleID;
        int Desired_Student_Grade;
        int Desired_LessonBundle_Cost;
        double Desired_LessonBundle_Multiplier;
        int Desired_Grade_GradeFee;

        List<double> TotalBundleCost = new List<double>();

        bool[] WeekDays = new bool[5];
        DateTime CurrentStartDate;
        string CurrentStartDay;
        string SearchDay;

        int x = 0;
        int y = 0;
        SqlDataReader DataReader;

        SqlConnection connection;
        string connectionString;
        string Querystring;
```

```
bool ContainsNumbers;
List<bool> ValidFields = new List<bool>();
DateTime EnteredDate = new DateTime();
string SelectedDate;
bool SpacePresent;
int Spacelocation;
char[] StringChracters = new char[50];
bool containsLetters;
string FirstSection;
string SecondSection;
bool PresenceOfSymbols;
bool PresenceOfPunctuation;
int NumberOfBookedDays;
DateTime AppendDate;
int AppendID;
bool AppendAttend = false;

bool Presence_Number;
bool Presence_Letter;
bool Presence_Punctuation;
bool Presence_Symbol;

string Postcode_Value;
string Postcode_BT;
string Postcode_Number;
string Postcode_Letters;

string SubstringTest;
int ATCount;

int TotalFields;
int TotalInstruments;
int TotalGradeRange;
int TotalBundleRange;

int CalendarButton;

int TotalSelectedDays;

List<int> Grade_IDs = new List<int>();
List<int> Instrument_IDs = new List<int>();
List<string> Specialisation = new List<string>();
List<string> RoomTypes = new List<string>();
List<int> Room_IDs = new List<int>();

List<int> Students_StudentID = new List<int>();
List<string> Students_FirstName = new List<string>();
List<string> Students_Surname = new List<string>();
List<string> Students_FullName_AND_StudentID = new List<string>();

List<int> Teachers_TeacherID = new List<int>();
List<string> Teachers_FirstName = new List<string>();
List<string> Teachers_Surname = new List<string>();
List<string> Teachers_FullName_AND_TeacherID = new List<string>();

List<int> LessonBundle_BundleID = new List<int>();
List<string> LessonBundle_Name = new List<string>();
List<string> LessonBundle_Name_AND_BundleID = new List<string>();

List<string> TimeList = new List<string>();
```

```
List<int> ScheduledLessons_PurchaseID = new List<int>();
List<int> purchasedLessons_IDs = new List<int>();

List<int> EndDate_Calculatiior_ID = new List<int>();
List<DateTime> EndDate_Calculator_EndDate = new List<DateTime>();
List<string> EndDate_Calculator_BookedWeeks = new List<string>();
List<DateTime> EndDate_Calculator_Final = new List<DateTime>();
List<DateTime> EndDate_Calculator_StartDate = new List<DateTime>();

DateTime EndDateValue;

int Desired_StudentID;

List<string> ScheduleWeeks = new List<string>();

int ValidationCount;
bool Continue;
int NumberOfWeeks;

public frmAddField()
{
    InitializeComponent();
    connectionString =
ConfigurationManager.ConnectionStrings[ "frmSplash.Properties.Settings.PrivateTuitionConnectionString" ].ConnectionString;
}

private void frmAddField_Load(object sender, EventArgs e)
{
    Hide_Additional();
    Hide_Errors();
    Store_StudentInformation();
    Store_TeacherInforamtion();
    Store_BundleInformation();
    Store_ScheduleInformation();
    Store_PurchaseInformation();

    TotalInstruments = 20;
    TotalGradeRange = 80;
    TotalBundleRange = 100;

    pictureBox4.Size = new Size(817, 68);
    pictureBox3.Size = new Size(827, 80);
    pictureBox7.Size = new Size(839, 91);

    btnAddRecord.Location = new Point(45, 123);
    btnReturn.Location = new Point(606, 123);

    Specialisation.Add("String");
    Specialisation.Add("Keyboard");
    Specialisation.Add("Brass");
    Specialisation.Add("WoodWind");
    Specialisation.Add("Percussion");
    Specialisation.Add("Vocal");
    Specialisation.Sort();

    RoomTypes.Add("Class Room");
    RoomTypes.Add("Hall");
    RoomTypes.Add("Practice Room");
```

```
RoomTypes.Sort();

TimeList.Add("13:00");
TimeList.Add("13:30");
TimeList.Add("14:00");
TimeList.Add("14:30");
TimeList.Add("15:00");
TimeList.Add("15:30");
TimeList.Add("16:00");
TimeList.Add("16:30");
TimeList.Add("17:00");
TimeList.Add("17:30");
TimeList.Add("18:00");
TimeList.Add("18:30");
TimeList.Add("19:00");
TimeList.Add("19:30");
TimeList.Add("20:00");
TimeList.Add("20:30");

ScheduleWeeks.Add("5 Weeks");
ScheduleWeeks.Add("10 Weeks");
ScheduleWeeks.Add("15 Weeks");
ScheduleWeeks.Add("20 Weeks");
ScheduleWeeks.Add("30 Weeks");

if (GlobalVariables.Purpose == "Update")
{
    UpdateField_Display();
}

DisplayDetails();
AssignTotalFields();
HideExcessInformation();

}

// DISPLAY COMPONENTS IN RELATION TO SELECTED TABLE
public void DisplayDetails()
{
    if (GlobalVariables.PreviousForm == "StudentTable")
    {
        lblFieldData2.Text = "First Name:";
        lblFieldData3.Text = "Other Name(s):";
        lblFieldData4.Text = "Surname";
        lblFieldData5.Text = "Date Of Birth:";
        lblFieldData6.Text = "Address:";
        lblFieldData7.Text = "Town";
        lblFieldData8.Text = "PostCode";
        lblFieldData9.Text = "Contact Number:";
        lblFieldData10.Text = "Email Address";
        lblFieldData11.Text = "GradeID";
        lblFieldData12.Text = "InstrumentID";
        lblFieldData13.Text = "Tuition Fee Recieved:";

        TotalFields = 12;

        tbFieldInfo11.Hide();
        cbFieldInfo1.Show();
        cbFieldInfo1.Location = new Point(146, 495);

        tbFieldInfo12.Hide();
        cbFieldInfo2.Show();
    }
}
```

```
cbFieldInfo2.Location = new Point(146, 527);

tbFieldInfo13.Hide();
CheckBox1.Show();
CheckBox1.Location = new Point(152, 559);

btnCalender.Show();
btnCalender.Location = new Point(308, 297);

tbFieldInfo2.MaxLength = 50;
tbFieldInfo3.MaxLength = 50;
tbFieldInfo4.MaxLength = 50;
tbFieldInfo5.MaxLength = 10;
tbFieldInfo6.MaxLength = 50;
tbFieldInfo7.MaxLength = 50;
tbFieldInfo8.MaxLength = 7;
tbFieldInfo9.MaxLength = 11; // Without first 3 digits
tbFieldInfo10.MaxLength = 50;

// Grade ID
// Fill Combo box
// This function will be used to store various pieces of information
from the schedule Table. This will be used to complete the schedule
using (connection = new SqlConnection(connectionString))
{
    connection.Open();
    using (SqlCommand Command = new SqlCommand("SELECT * FROM Grade",
connection))
    {
        DataReader = Command.ExecuteReader();
        while (DataReader.Read())
        {
            // For each record, store the corresponding piece of
information.
            Grade_ID.Add(DataReader.GetInt32(0));
        }
    }
    connection.Close();
}

cbFieldInfo1.DataSource = Grade_ID;
cbFieldInfo1.DropDownStyle = ComboBoxStyle.DropDownList;

// Instrument ID
// Fill Combo box
// This function will be used to store various pieces of information
from the schedule Table. This will be used to complete the schedule
using (connection = new SqlConnection(connectionString))
{
    connection.Open();
    using (SqlCommand Command = new SqlCommand("SELECT * FROM
Instruments", connection))
    {
        DataReader = Command.ExecuteReader();
        while (DataReader.Read())
        {
            // For each record, store the corresponding piece of
information.
            Instrument_IDS.Add(DataReader.GetInt32(0));
        }
    }
}
```

```
        connection.Close();
    }

    cbFieldInfo2.DataSource = Instrument_IDs;
    cbFieldInfo2.DropDownStyle = ComboBoxStyle.DropDownList;

    tbFieldInfo5.Enabled = false;

    lbMain.Text = "Add Student";
    btnAddRecord.Text = "Add New Student";
    btnReturn.Text = "Return to Students Table";

}

else if (GlobalVariables.PreviousForm == "TeacherTable")
{
    lblFieldData2.Text = "First Name:";
    lblFieldData3.Text = "Surname:";
    lblFieldData4.Text = "Address:";
    lblFieldData5.Text = "Town:";
    lblFieldData6.Text = "PostCode:";
    lblFieldData7.Text = "Email Address:";
    lblFieldData8.Text = "Contact Number:";
    lblFieldData9.Text = "Specialisation:";
    lblFieldData10.Text = "RoomID:";

    TotalFields = 9;

    tbFieldInfo10.Hide();
    tbFieldInfo9.Hide();
    cbFieldInfo2.Show();
    cbFieldInfo1.Show();
    cbFieldInfo1.Location = new Point(146, 431);
    cbFieldInfo2.Location = new Point(146, 463);

    cbFieldInfo1.DropDownStyle = ComboBoxStyle.DropDownList;
    cbFieldInfo2.DropDownStyle = ComboBoxStyle.DropDownList;

    cbFieldInfo1.DataSource = Specialisation;

    using (connection = new SqlConnection(connectionString))
    {
        connection.Open();
        using (SqlCommand Command = new SqlCommand("SELECT * FROM Room",
connection))
        {
            DataReader = Command.ExecuteReader();
            while (DataReader.Read())
            {
                // For each record, store the corresponding piece of
                information.
                Room_IDs.Add(DataReader.GetInt32(0));
            }
        }
        connection.Close();
    }

    cbFieldInfo2.DataSource = Room_IDs;
```

```
tbFieldInfo2.MaxLength = 50;
tbFieldInfo3.MaxLength = 50;
tbFieldInfo4.MaxLength = 60;
tbFieldInfo5.MaxLength = 60;
tbFieldInfo6.MaxLength = 7;
tbFieldInfo7.MaxLength = 60;
tbFieldInfo8.MaxLength = 11;
tbFieldInfo9.MaxLength = 40;

lbMain.Text = "Add Teacher";
btnAddRecord.Text = "Add New Teacher";
btnReturn.Text = "Return to Teacher Table";

QueryString = "INSERT INTO Teachers VALUES(@First_Name, @Surname,
@Address, @Town, @PostCode, @Email_Address, @Contact_Number, @Specialisation,
@RoomID";
}

else if (GlobalVariables.PreviousForm == "InstrumentTable")
{
    lblFieldData2.Text = "Instrument Type:";
    lblFieldData3.Text = "Instrument Name:";
    lblFieldData4.Text = "Quantity:";

    TotalFields = 3;

    tbFieldInfo2.Hide();
    cbFieldInfo1.Show();
    cbFieldInfo1.Location = new Point(146, 207);

    cbFieldInfo1.DropDownStyle = ComboBoxStyle.DropDownList;
    cbFieldInfo1.DataSource = Specialisation;

    tbFieldInfo2.MaxLength = 50;
    tbFieldInfo3.MaxLength = 50;
    tbFieldInfo4.MaxLength = 2;

    lbMain.Text = "Add Instrument";
    btnAddRecord.Text = "Add New Instrument";
    btnReturn.Text = "Return to Instrument Table";

    Querystring = "INSERT INTO Instruments VALUES(@Instrument Type,
@Instrument Name, @Quantity";
}

else if (GlobalVariables.PreviousForm == "RoomTable")
{
    lblFieldData2.Text = "Room Type:";
    lblFieldData3.Text = "Specialisation:";

    TotalFields = 2;

    tbFieldInfo2.Hide();
    tbFieldInfo3.Hide();

    cbFieldInfo1.Show();
    cbFieldInfo2.Show();
    cbFieldInfo1.Location = new Point(146, 207);
    cbFieldInfo2.Location = new Point(146, 239);

    cbFieldInfo1.DropDownStyle = ComboBoxStyle.DropDownList;
    cbFieldInfo1.DataSource = RoomTypes;
```

```
cbFieldInfo2.DropDownStyle = ComboBoxStyle.DropDownList;
cbFieldInfo2.DataSource = Specialisation;

tbFieldInfo2.MaxLength = 25;
tbFieldInfo3.MaxLength = 50;

lbMain.Text = "Add Room";
btnAddRecord.Text = "Add New Room";
btnReturn.Text = "Return to Room Table";

Querystring = "INSERT INTO Room VALUES(@Room_Type, @Specialisation";
}

else if (GlobalVariables.PreviousForm == "GradeTable")
{
    lblFieldData2.Text = "Grade Level:";
    lblFieldData3.Text = "GradeFee:";

    TotalFields = 2;

    tbFieldInfo2.MaxLength = 25;
    tbFieldInfo3.MaxLength = 3;

    lbMain.Text = "Add Grade";
    btnAddRecord.Text = "Add New Grade";
    btnReturn.Text = "Return to Grade Table";

    Querystring = "INSERT INTO Grade VALUES(@GradeLevel, @GradeFee";
}

else if (GlobalVariables.PreviousForm == "LessonBundleTable")
{
    lblFieldData2.Text = "Lesson Bundle:";
    lblFieldData3.Text = "Bundle Cost:";
    lblFieldData4.Text = "Multiplier:";

    TotalFields = 3;

    tbFieldInfo2.MaxLength = 40;
    tbFieldInfo3.MaxLength = 3;
    tbFieldInfo4.MaxLength = 3;

    lbMain.Text = "New Lesson Bundle:";
    btnAddRecord.Text = "Add New Lesson Bundle";
    btnReturn.Text = "Return to Lesson Bundles Table";

    Querystring = "INSERT INTO LessonBundles VALUES(@Lesson Bundle, Bundle
Cost, Multiplier (Discount Rate))";
}

else if (GlobalVariables.PreviousForm == "ScheduledLessonTable" ||
GlobalVariables.PreviousForm == "Calender")
{
    lblFieldData2.Text = "Student ID:";
    lblFieldData3.Text = "Teacher ID:";
    lblFieldData4.Text = "Purchased Lesson:";
    lblFieldData5.Text = "No. of weeks:";
    lblFieldData6.Text = "Start Date:";
    lblFieldData7.Text = "Booked Time:";
    lblFieldData8.Text = "Booked Day(s):";
```

```
TotalFields = 8;

tbFieldInfo2.Hide();
tbFieldInfo3.Hide();
//tbFieldInfo4.Hide();
tbFieldInfo7.Hide();

if (GlobalVariables.Purpose == "Add")
{
    tbFieldInfo9.Hide();
}

cbFieldInfo1.Show();
cbFieldInfo2.Show();
cbFieldInfo3.Show();
cbFieldInfo4.Show();

tbFieldInfo9.Enabled = false;
tbFieldInfo6.Enabled = false;
tbFieldInfo4.Enabled = false;

btnCalender.Location = new Point(308, 325);
btnDataGrid.Location = new Point(308, 265);
btnCalender.Show();
btnDataGrid.Show();

btnTimeSelection.Location = new Point(308, 390);
btnTimeSelection.Show();

cbFieldInfo1.Location = new Point(146, 207);
cbFieldInfo2.Location = new Point(146, 239);
cbFieldInfo3.Location = new Point(146, 303);
cbFieldInfo4.Location = new Point(146, 367);

cbFieldInfo1.DropDownStyle = ComboBoxStyle.DropDownList;
cbFieldInfo2.DropDownStyle = ComboBoxStyle.DropDownList;
cbFieldInfo3.DropDownStyle = ComboBoxStyle.DropDownList;
cbFieldInfo4.DropDownStyle = ComboBoxStyle.DropDownList;

cbFieldInfo1.DataSource = Students_FullName_AND_StudentID;
cbFieldInfo2.DataSource = Teachers_FullName_AND_TeacherID;
cbFieldInfo3.DataSource = ScheduleWeeks;
cbFieldInfo4.DataSource = TimeList;

tbFieldInfo5.MaxLength = 20;
tbFieldInfo6.MaxLength = 10;
tbFieldInfo7.MaxLength = 100;
tbFieldInfo8.MaxLength = 7;

lbMain.Text = "Schedule New Lesson";
btnAddRecord.Text = "Schedule New Lesson";
btnReturn.Text = "Return to Scheduled Lessons Table";

Querystring = "INSERT INTO Scheduled_Lessons VALUES(@StudentID,
@TeacherID, @LessonBundleID, @Number_Of_Weeks, @Start_Date, @Booked_Day(s),
@Booked_Time, @End-Date)";
}

else if (GlobalVariables.PreviousForm == "PurchasedLessonBundleTable")
{
    lblFieldData2.Text = "Student ID:";
```

```
lblFieldData3.Text = "Purchase ID:";  
lblFieldData4.Text = "Purchase Date:";  
lblFieldData5.Text = "Payment Method:";  
lblFieldData6.Text = "Payment Received:";  
lblFieldData7.Text = "payment Receipt Date:";  
lblFieldData8.Text = "Bundle Cost:";  
  
TotalFields = 7;  
  
tbFieldInfo2.Hide();  
tbFieldInfo3.Hide();  
tbFieldInfo8.Hide();  
lblFieldData8.Hide();  
  
cbFieldInfo1.Show();  
cbFieldInfo2.Show();  
  
cbFieldInfo1.Location = new Point(146, 207);  
cbFieldInfo2.Location = new Point(146, 239);  
  
tbFieldInfo4.MaxLength = 10;  
tbFieldInfo5.MaxLength = 25;  
tbFieldInfo6.MaxLength = 1;  
tbFieldInfo7.MaxLength = 10;  
  
lbMain.Text = "New Lesson Purchased:";  
btnAddRecord.Text = "Purchase New Lesson";  
btnReturn.Text = "Return to Purchased Lessons Table";  
  
Querystring = "INSERT INTO Scheduled_Lessons VALUES(@StudentID,  
@LessonBundleID, @Purchase_Date, @Payment_Method, @Payment-Received, @Payment_Received  
Date, @Bundle_Cost(After_Discount))";  
}  
}  
  
public void Data_Validation()  
{  
    ValidationCount = 0;  
    Continue = false;  
    for (int x = 0; x < ValidFields.Count(); x++)  
    {  
        if (ValidFields[x] == true)  
        {  
            ValidationCount++;  
        }  
    }  
  
    if (ValidationCount == ValidFields.Count())  
    {  
        AddNewInformation();  
    }  
}  
  
public void AddNewInformation()  
{  
    if (GlobalVariables.PreviousForm == "StudentTable")  
    {  
        if (GlobalVariables.Purpose == "Add")  
        {  
    }
```

```
Querystring = "INSERT INTO Students VALUES(@First_Name,
@Other_Names, @Surname, @DOB, @Address, @Town, @post, @Contact, @Email, @Grade,
@Instrument, @Tuition)";
}
else
{
    Querystring = "UPDATE Students SET First_Name=@First_Name,
Other_Names=@Other_Names, Surname=@Surname, DateOfBirth=@DOB, Address=@Address,
Town=@Town, PostCode=@post, Contact_Number=@Contact, Email_Address=@Email,
GradeID=@Grade, InstrumentID=@Instrument, Tuition_Fee_Received=@Tuition WHERE
StudentID=@StudentID";
}

using (connection = new SqlConnection(connectionString))
{
    connection.Open();
    using (SqlCommand Command = new SqlCommand(Querystring,
connection))
    {
        Command.Parameters.AddWithValue("@StudentID",
tbFieldInfo14.Text);
        Command.Parameters.AddWithValue("@First_Name",
tbFieldInfo2.Text);
        Command.Parameters.AddWithValue("@Other_Names",
tbFieldInfo3.Text);
        Command.Parameters.AddWithValue("@Surname",
tbFieldInfo4.Text);
        Command.Parameters.AddWithValue("@DOB",
Convert.ToDateTime(tbFieldInfo5.Text));
        Command.Parameters.AddWithValue("@Address",
tbFieldInfo6.Text);
        Command.Parameters.AddWithValue("@Town", tbFieldInfo7.Text);
        Command.Parameters.AddWithValue("@post", tbFieldInfo8.Text);
        Command.Parameters.AddWithValue("@Contact",
tbFieldInfo9.Text);
        Command.Parameters.AddWithValue("@Email", tbFieldInfo10.Text);
        Command.Parameters.AddWithValue("@Grade",
Convert.ToInt32(cbFieldInfo1.Text));
        Command.Parameters.AddWithValue("@Instrument",
Convert.ToInt32(cbFieldInfo2.Text));
        Command.Parameters.AddWithValue("@Tuition", CheckBox1.Text);

        Command.ExecuteNonQuery();
    }
    connection.Close();
}
}

else if (GlobalVariables.PreviousForm == "TeacherTable")
{
    if (GlobalVariables.Purpose == "Add")
    {

    }
    else
    {

    }
}

else if (GlobalVariables.PreviousForm == "InstrumentTable")
```

```
{  
    if (GlobalVariables.Purpose == "Add")  
    {  
    }  
    else  
    {  
    }  
}  
  
else if (GlobalVariables.PreviousForm == "RoomTable")  
{  
    if (GlobalVariables.Purpose == "Add")  
    {  
    }  
    else  
    {  
    }  
}  
  
else if (GlobalVariables.PreviousForm == "GradeTable")  
{  
    if (GlobalVariables.Purpose == "Add")  
    {  
    }  
    else  
    {  
    }  
}  
  
else if (GlobalVariables.PreviousForm == "LessonBundleTable")  
{  
    if (GlobalVariables.Purpose == "Add")  
    {  
    }  
    else  
    {  
    }  
}  
  
else if (GlobalVariables.PreviousForm == "ScheduledLessonTable" ||  
GlobalVariables.PreviousForm == "Calender")  
{  
    if (GlobalVariables.Purpose == "Add")  
    {  
    }  
    else  
    {  
    }  
}  
  
}
```

```
}

public void AssignTotalFields()
{
    for (int x = 0; x < TotalFields; x++)
    {
        ValidFields.Add(false);
    }
}

public void Store_ScheduleInformation()
{
    using (connection = new SqlConnection(connectionString))
    {
        connection.Open();
        using (SqlCommand Command = new SqlCommand("SELECT * FROM
Scheduled_Lessons", connection))
        {
            DataReader = Command.ExecuteReader();
            while (DataReader.Read())
            {
                // For each record, store the corresponding piece of
information.
                ScheduledLessons_PurchaseID.Add(DataReader.GetInt32(2));

                EndDate_Calculator_ID.Add(DataReader.GetInt32(0));
                EndDate_Calculator_BookedWeeks.Add(DataReader.GetString(4));
                EndDate_Calculator_EndDate.Add(DataReader.GetDateTime(8));
                EndDate_Calculator_StartDate.Add(DataReader.GetDateTime(5));
            }
        }
        connection.Close();
    }
}

public void Store_PurchaseInformation()
{
    using (connection = new SqlConnection(connectionString))
    {
        connection.Open();
        using (SqlCommand Command = new SqlCommand("SELECT * FROM
LessonsPurchased", connection))
        {
            DataReader = Command.ExecuteReader();
            while (DataReader.Read())
            {
                // For each record, store the corresponding piece of
information.
                purchasedLessons_IDs.Add(DataReader.GetInt32(0));
            }
            DataReader.Close();

            using (SqlCommand Cmd2 = new SqlCommand("SELECT * FROM
LessonsPurchased WHERE StudentID = @StudentID AND Scheduled = @Scheduled",
connection))
            {
                Cmd2.Parameters.AddWithValue("@StudentID", Desired_StudentID);
                Cmd2.Parameters.AddWithValue("@Scheduled", 0);
            }
        }
    }
}
```

```
//Cmd2.Parameters.AddWithValue("@StudentID",
Students_StudentID[cbFieldInfo1.SelectedIndex]);  
  
        using (SqlDataAdapter Adaptor = new SqlDataAdapter())
{
    DataTable PurchasedLessons = new DataTable();
    Adaptor.SelectCommand = Cmd2;
    Adaptor.Fill(PurchasedLessons);  
  
    DataGrid_PurchasedLessons.DataSource = PurchasedLessons;
}
}  
connection.Close();  
}  
}  
  
public void Store_StudentInformation()
{
    using (connection = new SqlConnection(connectionString))
    {
        connection.Open();
        using (SqlCommand Command = new SqlCommand("SELECT * FROM Students",
connection))
        {
            DataReader = Command.ExecuteReader();
            while (DataReader.Read())
            {
                // For each record, store the corresponding piece of
information.  
                Students_StudentID.Add(DataReader.GetInt32(0));
                Students_FirstName.Add(DataReader.GetString(1));
                Students_Surname.Add(DataReader.GetString(3));
            }
        }
        connection.Close();
    }
}  
  
for (int x = 0; x < Students_StudentID.Count(); x++)
{
    Students_FullName_AND_StudentID.Add(Students_StudentID[x] + " |
" + Students_FirstName[x] + " " + Students_Surname[x]);
}
}  
  
public void Store_TeacherInformation()
{
    using (connection = new SqlConnection(connectionString))
    {
        connection.Open();
        using (SqlCommand Command = new SqlCommand("SELECT * FROM Teachers",
connection))
        {
            DataReader = Command.ExecuteReader();
            while (DataReader.Read())
            {
                // For each record, store the corresponding piece of
information.  
                Teachers_TeacherID.Add(DataReader.GetInt32(0));
                Teachers_FirstName.Add(DataReader.GetString(1));
                Teachers_Surname.Add(DataReader.GetString(2));
            }
        }
    }
}
```

```
        }
        connection.Close();
    }

    for (int x = 0; x < Teachers_TeacherID.Count(); x++)
    {
        Teachers_FullName_AND_TeacherID.Add(Teachers_TeacherID[x] + " | "
" + Teachers_FirstName[x] + " " + Teachers_Surname[x]);
    }
}

public void Store_BundleInformation()
{
    using (connection = new SqlConnection(connectionString))
    {
        connection.Open();
        using (SqlCommand Command = new SqlCommand("SELECT * FROM
LessonBundles", connection))
        {
            DataReader = Command.ExecuteReader();
            while (DataReader.Read())
            {
                // For each record, store the corresponding piece of
information.
                LessonBundle_ID.Add(DataReader.GetInt32(0));
                LessonBundle_Name.Add(DataReader.GetString(1));
            }
        }
        connection.Close();
    }

    for (int x = 0; x < LessonBundle_ID.Count(); x++)
    {
        LessonBundle_Name_AND_BundleID.Add(LessonBundle_ID[x] + " | " +
LessonBundle_Name[x]);
    }
}

// DISPLAY UPDATE RECORD
public void UpdateField_Display()
{
    pictureBox4.Size = new Size(817, 40);
    pictureBox3.Size = new Size(827, 50);
    pictureBox7.Size = new Size(839, 60);

    btnAddRecord.Location = new Point(46, 107);
    btnReturn.Location = new Point(606, 107);

    lblFieldData14.Show();
    lblFieldData14.Location = new Point(25, 175);

    tbFieldInfo14.Show();
    tbFieldInfo14.Location = new Point(146, 174);

    tbFieldInfo14.Text = GlobalVariables.UpdateID.ToString();

    if (GlobalVariables.PreviousForm == "StudentTable")
    {
        for (int x = 0; x < GlobalVariables.Student_PostCode.Length; x++)
        {
            string test9 = GlobalVariables.Student_PostCode.Substring(x, 1);
```

```
        if (test9 == " ")
    {
        string FirstHalf =
GlobalVariables.Student_PostCode.Substring(0, x);
        string SecondHalf =
GlobalVariables.Student_PostCode.Substring(x + 1,
(GlobalVariables.Student_PostCode.Length - (x + 1)));
        GlobalVariables.Student_PostCode = String.Format("{0}{1}",
FirstHalf, SecondHalf);
        break;
    }
}

lbMain.Text = "Update Student Record";
lblFieldData14.Text = "Student ID:";
tbFieldInfo2.Text = GlobalVariables.Student_FirstName;
tbFieldInfo3.Text = GlobalVariables.Student_OtherNames;
tbFieldInfo4.Text = GlobalVariables.Student_Surname;
tbFieldInfo5.Text = GlobalVariables.Student_DOB;
tbFieldInfo6.Text = GlobalVariables.Student_Address;
tbFieldInfo7.Text = GlobalVariables.Student_Town;
tbFieldInfo8.Text = GlobalVariables.Student_PostCode;
tbFieldInfo9.Text = GlobalVariables.Student_ContactNumber.ToString();
tbFieldInfo10.Text = GlobalVariables.Student_Email;
cbFieldInfo1.Text = GlobalVariables.Student_GradeID.ToString();
cbFieldInfo2.Text = GlobalVariables.Student_InstrumentID.ToString();
if (GlobalVariables.Student_PaymentFee == "True")
{
    CheckBox1.CheckState = CheckState.Checked;
}
else
{
    CheckBox1.CheckState = CheckState.Unchecked;
}
}

else if (GlobalVariables.PreviousForm == "TeacherTable")
{
    for (int x = 0; x < GlobalVariables.Teacher_PostCode.Length; x++)
    {
        string test9 = GlobalVariables.Teacher_PostCode.Substring(x, 1);
        if (test9 == " ")
        {
            string FirstHalf =
GlobalVariables.Teacher_PostCode.Substring(0, x);
            string SecondHalf =
GlobalVariables.Teacher_PostCode.Substring(x + 1,
(GlobalVariables.Teacher_PostCode.Length - (x + 1)));
            GlobalVariables.Teacher_PostCode = String.Format("{0}{1}",
FirstHalf, SecondHalf);
            break;
        }
    }

    lbMain.Text = "Update Teacher Recrod";
    lblFieldData14.Text = "Teacher ID";
    tbFieldInfo2.Text = GlobalVariables.Teacher_FirstName;
    tbFieldInfo3.Text = GlobalVariables.Teacher_Surname;
    tbFieldInfo4.Text = GlobalVariables.Teacher_Address;
    tbFieldInfo5.Text = GlobalVariables.Teacher_Town;
    tbFieldInfo6.Text = GlobalVariables.Teacher_PostCode;
    tbFieldInfo7.Text = GlobalVariables.Teacher_Email;
```

```
        tbFieldInfo8.Text = GlobalVariables.Teacher_ContactNumber;
        tbFieldInfo9.Text = GlobalVariables.Teacher_Specialisation;
        cbFieldInfo1.Text = GlobalVariables.Teacher_RoomID;
    }

    else if (GlobalVariables.PreviousForm == "InstrumentTable")
    {
        lbMain.Text = "Update Instrument Recrod";
        lblFieldData14.Text = "Instrument ID";
        cbFieldInfo1.Text = GlobalVariables.Instrument_Type;
        tbFieldInfo3.Text = GlobalVariables.Instrument_Name;
        tbFieldInfo4.Text = GlobalVariables.Instrument_Quantity;
    }

    else if (GlobalVariables.PreviousForm == "RoomTable")
    {
        lbMain.Text = "Update Room Recrod";
        lblFieldData14.Text = "Room ID";
        tbFieldInfo2.Text = GlobalVariables.Room_Type;
        cbFieldInfo1.Text = GlobalVariables.Room_Specialisation;
    }

    else if (GlobalVariables.PreviousForm == "GradeTable")
    {
        lbMain.Text = "Update Grade Recrod";
        lblFieldData14.Text = "Grade ID";
        tbFieldInfo2.Text = GlobalVariables.Grade_Level;
        tbFieldInfo3.Text = GlobalVariables.Grade_Fee;
    }

    else if (GlobalVariables.PreviousForm == "LessonBundleTable")
    {
        lbMain.Text = "Update LessonBundle Recrod";
        lblFieldData14.Text = "Bundle ID";
        tbFieldInfo2.Text = GlobalVariables.Bundle_Name;
        tbFieldInfo3.Text = GlobalVariables.Bundle_Cost;
        tbFieldInfo4.Text = GlobalVariables.Bundle_Discount;
    }

    else if (GlobalVariables.PreviousForm == "ScheduledLessonTable" ||
GlobalVariables.PreviousForm == "Calender")
    {

        tbFieldInfo9.Show();
        lblFieldData9.Show();
        lblFieldData9.Text = "End Date:";

        btnCalendar2.Location = new Point(308, 421);

        lbMain.Text = "Update Scheduled Lesson Recrod";
        lblFieldData14.Text = "Schedule ID";
        cbFieldInfo1.Text = GlobalVariables.Schedule_StudentID;
        cbFieldInfo2.Text = GlobalVariables.Schedule_TeacherID;
        tbFieldInfo4.Text = GlobalVariables.Schedule_PurchaseID;
        tbFieldInfo5.Text = GlobalVariables.Schedule_Weeks;
        tbFieldInfo6.Text = GlobalVariables.Schedule_StartDate;
        tbFieldInfo8.Text = GlobalVariables.Schedule_BookedDays;
        cbFieldInfo4.Text = GlobalVariables.Schedule_BookedTime;
        tbFieldInfo9.Text = GlobalVariables.Schedule_EndDate;

    }
}
```

```
        else if (GlobalVariables.PreviousForm == "PurchasedLessonBundleTable")
    {
        lbMain.Text = "Update Purchased Lesson Record";
        lblFieldData14.Text = "Purchase ID";
        cbFieldInfo1.Text = GlobalVariables.Purchase_StudentID;
        cbFieldInfo2.Text = GlobalVariables.Purchase_BundleID;
        tbFieldInfo4.Text = GlobalVariables.Purchase_Date;
        tbFieldInfo5.Text = GlobalVariables.Purchase_Method;
        tbFieldInfo6.Text = GlobalVariables.Purchase_Received;
        tbFieldInfo7.Text = GlobalVariables.Purchase_ReceivedDate;
        tbFieldInfo8.Text = GlobalVariables.Purchase_BundleCosts;
    }
}

// RETURN FUNCTIONS
private void btnReturn_Click(object sender, EventArgs e)
{
    if (GlobalVariables.PreviousForm == "StudentTable")
    {
        frmStudents StudentTable = new frmStudents();
        StudentTable.Show();
        this.Hide();
    }
    else if (GlobalVariables.PreviousForm == "TeacherTable")
    {
        frmTeachers TeacherTable = new frmTeachers();
        TeacherTable.Show();
        this.Hide();
    }
    else if (GlobalVariables.PreviousForm == "InstrumentTable")
    {
        frmInstrument InstrumentTable = new frmInstrument();
        InstrumentTable.Show();
        this.Hide();
    }
    else if (GlobalVariables.PreviousForm == "RoomTable")
    {
        frmRoom RoomTable = new frmRoom();
        RoomTable.Show();
        this.Hide();
    }
    else if (GlobalVariables.PreviousForm == "GradeTable")
    {
        frmGrade GradeTable = new frmGrade();
        GradeTable.Show();
        this.Hide();
    }
    else if (GlobalVariables.PreviousForm == "LessonBundleTable")
    {
        frmLessonBundle LessonBundleTable = new frmLessonBundle();
        LessonBundleTable.Show();
        this.Hide();
    }
    else if (GlobalVariables.PreviousForm == "ScheduledLessonTable" ||
GlobalVariables.PreviousForm == "Calender")
    {
        frmScheduleTable Scheduled_Lessons = new frmScheduleTable();
        Scheduled_Lessons.Show();
        this.Hide();
    }
}
```

```
        else if (GlobalVariables.PreviousForm == "PurchasedLessonBundleTable")
    {
        PurchasedLessonBundles Purchased_Lessons = new
PurchasedLessonBundles();
        Purchased_Lessons.Show();
        this.Hide();
    }

    lblFieldData2.Show();
    lblFieldData3.Show();
    lblFieldData4.Show();
    lblFieldData5.Show();
    lblFieldData6.Show();
    lblFieldData7.Show();
    lblFieldData8.Show();
    lblFieldData9.Show();
    lblFieldData10.Show();
    lblFieldData11.Show();
    lblFieldData12.Show();
    lblFieldData13.Show();
}

// ADD FUNCTION
private void btnAddRecord_Click(object sender, EventArgs e)
{
    lblErrorPlaceholder2.Hide();
    lblErrorPlaceholder3.Hide();
    lblErrorPlaceholder4.Hide();
    lblErrorPlaceholder5.Hide();
    lblErrorPlaceholder6.Hide();
    lblErrorPlaceholder7.Hide();
    lblErrorPlaceholder8.Hide();
    lblErrorPlaceholder9.Hide();
    lblErrorPlaceholder10.Hide();
    lblErrorPlaceholder11.Hide();
    lblErrorPlaceholder12.Hide();
    lblErrorPlaceholder13.Hide();

    // Clear all lists in advance, as to reduce data redundancy.
    AllScheduledLessonsIDs.Clear();
    AllScheduledLessonsStudentIDs.Clear();
    AllScheduledLessonsTeacherIDs.Clear();
    AllScheduledLessonsBundleIDs.Clear();
    AllScheduledBookedDays.Clear();
    AllScheduledLessonsStartDates.Clear();
    AllScheduledLessonsEndDates.Clear();
    AllStartWeekDays.Clear();
    AllLessonDatesScheduleIDs.Clear();
    PurchasedLesson_IDs.Clear();
    PurchasedLessons_StudentID.Clear();
    PurchasedLessons_LessonBundleID.Clear();
    Student_ID.Clear();
    Student_GradeID.Clear();
    LessonBundle_ID.Clear();
    LessonBundle_BundleCost.Clear();
    LessonBundle_Multiplier.Clear();
    Grade_ID.Clear();
    Grade_Cost.Clear();

    if (GlobalVariables.PreviousForm == "StudentTable")
    {
        Validation_Student();
    }
}
```

```
        }
        if (GlobalVariables.PreviousForm == "TeacherTable")
        {
            Validation_Teacher();
        }
        if (GlobalVariables.PreviousForm == "InstrumentTable")
        {
            Validation_Instrument();
        }
        if (GlobalVariables.PreviousForm == "RoomTable")
        {
            Validation_Room();
        }
        if (GlobalVariables.PreviousForm == "GradeTable")
        {
            Validation_Grade();
        }
        if (GlobalVariables.PreviousForm == "LessonBundleTable")
        {
            Validation_LessonBundle();
        }
        if (GlobalVariables.PreviousForm == "PurchasedLessonBundleTable")
        {
            StoreInformation_BundleCalculator();
            BundleCostCalculator();
        }
        if (GlobalVariables.PreviousForm == "ScheduledLessonTable" ||
GlobalVariables.PreviousForm == "Calender")
        {
            Validation_ScheduledLesson();
            Data_Validation();

            // Details are being stored locally.
            LocalInformationStorage();

            // This block of code is used to remove any schedules which have
already been added.
            try
            {
                for (int T = 0; T < AllScheduledLessonsIDs.Count(); T++)
                {
                    for (int X = 0; X < AllLessonDatesScheduleIDs.Count(); X++)
                    {
                        if (AllScheduledLessonsIDs[T] ==
AllLessonDatesScheduleIDs[X])
                        {
                            AllScheduledLessonsBundleIDs.RemoveAt(x);
                            AllScheduledBookedDays.RemoveAt(x);
                            AllScheduledLessonsEndDates.RemoveAt(x);
                            AllScheduledLessonsIDs.RemoveAt(x);
                            AllScheduledLessonsStartDates.RemoveAt(x);
                            AllScheduledLessonsStudentIDs.RemoveAt(x);
                            AllScheduledLessonsTeacherIDs.RemoveAt(x);
                            x -= 1;
                        }
                    }
                }
            }
            catch (Exception ex)
            {
```

```
    }

    for (int x = 0; x < AllScheduledLessonsIDs.Count(); x++)
    {
        for (int y = 0; y < LessonDates_Archive_ID.Count(); y++)
        {
            if (AllScheduledLessonsIDs[x] == LessonDates_Archive_ID[y])
            {
                AllScheduledLessonsBundleIDs.RemoveAt(x);
                AllScheduledBookedDays.RemoveAt(x);
                AllScheduledLessonsEndDates.RemoveAt(x);
                AllScheduledLessonsIDs.RemoveAt(x);
                AllScheduledLessonsStartDates.RemoveAt(x);
                AllScheduledLessonsStudentIDs.RemoveAt(x);
                AllScheduledLessonsTeacherIDs.RemoveAt(x);
            }
        }
    }

    // Identify Booked Days Against Dates
    for (int T = 0; T < AllScheduledLessonsIDs.Count(); T++)
    {
        // This passes each booked lessons value into the method
        // individually. This should identify booked days.
        ScheduledDays(AllScheduledBookedDays[T]);
        CurrentStartDate = AllScheduledLessonsStartDates[T];
        CurrentStartDay = Convert.ToString(CurrentStartDate.DayOfWeek);

        // This will identify if the user has booked two days.
        if (NumberOfBookedDays == 2)
        {
            // If so, both this block of code will run twice, to handle
            // each day individually.
            for (int R = 0; R < 2; R++)
            {
                // This block of code will be used to identify the first
                // booked day which has been selected.
                if (WeekDays[0] == true)
                {
                    SearchDay = "Monday"; // This assigns the value which
                    // will be compared.
                    WeekDays[0] = false; // The associated bool is then
                    // set to false, as to ensure that the next booked day will be used, on the next round.
                }

                if (WeekDays[1] == true)
                {
                    SearchDay = "Tuesday";
                    WeekDays[1] = false;
                }

                if (WeekDays[2] == true)
                {
                    SearchDay = "Wednesday";
                    WeekDays[2] = false;
                }

                if (WeekDays[3] == true)
                {

```

```
        SearchDay = "Thursday";
        WeekDays[3] = false;
    }

    if (WeekDays[4] == true)
    {
        SearchDay = "Friday";
        WeekDays[4] = false;
    }

    AppendID = AllScheduledLessonsIDs[T];
    Day_Date_Comparision(SearchDay, CurrentStartDate,
AllScheduledLessonsEndDates[T]);
}
// This block of code will run if the current record has one
booked day.
else if (NumberOfBookedDays == 1)
{
    // This nested if statement will identify the booked day, and
assign the corriponding value.
    if (WeekDays[0] == true)
    {
        SearchDay = "Monday"; // This assigns the value which will
be compared.
    }
    else if (WeekDays[1] == true)
    {
        SearchDay = "Tuesday";
    }
    else if (WeekDays[2] == true)
    {
        SearchDay = "Wednesday";
    }
    else if (WeekDays[3] == true)
    {
        SearchDay = "Thursday";
    }
    else if (WeekDays[4] == true)
    {
        SearchDay = "Friday";
    }
    else
    {
        // In the off chance that the value is not found within
the nested if statement, this place holder error message should appear.
        MessageBox.Show("Incorrect Value entered: BOOKED DAYS");
    }
}

// Try was implemented due to the off chance that a value may
not be located within the IF block, thus this is a precaution to ensure the system
stays operational.
try
{
    // Assign current Schedule ID value.
    AppendID = AllScheduledLessonsIDs[T];
    // Pass through booked day value, for comparision and
eventually appending.
    Day_Date_Comparision(SearchDay, CurrentStartDate,
AllScheduledLessonsEndDates[T]);
}
catch (Exception ex)
```

```
        {
            // If an error becomes apparent, simply break from the
            // system, and display error message.
            MessageBox.Show("Could not complete operation");
            break;
        }

    }
else
{
    MessageBox.Show("Too many days have been booked for a
lesson");
}

}

for (int x = 0; x < EndDate_Calculator_ID.Count(); x++)
{
    for (int y = 0; y < EndDate_Calculator_BookedWeeks[x].Length; y++)
    {
        string Sub = EndDate_Calculator_BookedWeeks[x].Substring(y,
1);
        if (Sub == " ")
        {
            NumberOfWeeks =
Convert.ToInt32(EndDate_Calculator_BookedWeeks[x].Substring(0, y));
            break;
        }
    }

    EndDateValue = EndDate_Calculator_StartDate[x];
    for (int z = 0; z < NumberOfWeeks; z++)
    {
        EndDateValue = EndDateValue.AddDays(7);
    }

    using (connection = new SqlConnection(connectionString))
    {
        connection.Open();
        using (SqlCommand Command = new SqlCommand("UPDATE
Scheduled_Lessons SET End_Date=@EndDate WHERE ScheduleID=@ID", connection))
        {
            Command.Parameters.AddWithValue("@EndDate", EndDateValue);
            Command.Parameters.AddWithValue("@ID",
EndDate_Calculator_ID[x]);
            Command.ExecuteNonQuery();
        }
        connection.Close();
    }
}
}
```

```
// DATA VALIDATION
public void Validation_Student()
{
    // Sets bool true if number is located.
    ContainsNumbers = tbFieldInfo2.Text.Any(char.IsDigit);
    PresenceOfPunctuation = tbFieldInfo2.Text.Any(char.IsPunctuation);
    PresenceOfSymbols = tbFieldInfo2.Text.Any(char.IsSymbol);

    // FIRST NAME
    if (tbFieldInfo2.Text != "")
    {
        // If Field Contains a value. <Run If Statement>

        if (PresenceOfPunctuation == false && PresenceOfSymbols == false)
        {
            // If Field does not contain a non-numeric or alphabetical
            character. <Run If Statement>
            if (ContainsNumbers == true)
            {
                // If value contains numeric Characters. <Error Message>
                ValidFields[0] = false;
                lblErrorPlaceholder2.Show();
                lblErrorPlaceholder2.Text = "Please Remove Numerical
                Characters.";
            }
            else
            {
                // Value Contains solely text. <Validate Field>
                ValidFields[0] = true;
            }
        }
        else
        {
            // If field does contain non-Numeric or Alphabetical characters.
            <Error Message>
            ValidFields[0] = false;
            lblErrorPlaceholder2.Show();
            lblErrorPlaceholder2.Text = "Please Remove Symbols or
            Punctuation.";
        }
    }
    else
    {
        // If Textbox contains no value.
        ValidFields[0] = false;
        lblErrorPlaceholder2.Show();
        lblErrorPlaceholder2.Text = "Data Required. Please Complete.";
    }

    // OTHER NAME(S)
    ContainsNumbers = tbFieldInfo3.Text.Any(char.IsDigit);
    PresenceOfPunctuation = tbFieldInfo3.Text.Any(char.IsPunctuation);
    PresenceOfSymbols = tbFieldInfo3.Text.Any(char.IsSymbol);

    if (tbFieldInfo3.Text != "")
    {
        // If field contains a value. <Run If Statement>
```

```
        if (PresenceOfSymbols == false && PresenceOfPunctuation == false)
    {
        // If value does not contain non-alphabetic or numerical values.
<Run IF Statement>

        if (ContainsNumbers == true)
    {
        // If Value Contains Numerical characters. <Error Message>
        ValidFields[1] = false;
        lblErrorPlaceholder3.Show();
        lblErrorPlaceholder3.Text = "Please Remove Numerical
Characters.";
    }
    else
    {
        // If value contains no numerical characters or symbols.
<Validate Field>
        ValidFields[1] = true;
    }
}
else
{
    // If Value Contains symbol(s). <Error Message>
    ValidFields[1] = false;
    lblErrorPlaceholder3.Show();
    lblErrorPlaceholder3.Text = "Please remove symbols or
punctuation.";
}
}
else
{
    // If no value present. <Error Message>
    ValidFields[1] = false;
    lblErrorPlaceholder3.Show();
    lblErrorPlaceholder3.Text = "Data Required. please Complete.";
}

// SURNAME
ContainsNumbers = tbFieldInfo4.Text.Any(char.IsDigit);
PresenceOfPunctuation = tbFieldInfo4.Text.Any(char.IsPunctuation);
PresenceOfSymbols = tbFieldInfo4.Text.Any(char.IsSymbol);

if (tbFieldInfo4.Text != "")
{
    // If field contains a value. <Run If Statement>

    if (PresenceOfPunctuation == false && PresenceOfSymbols == false)
    {
        // Value does not contain symbols. <Run If Statement>

        if (ContainsNumbers == true)
    {
        // Value Contains Numbers. <Error Message>

        ValidFields[2] = false;
        lblErrorPlaceholder4.Show();
    }
}
```

```

        lblErrorPlaceholder4.Text = "Please Remove Numerical
Characters.";
    }
    else
    {
        // Value contains valid characters. <Validate Field>
        ValidFields[2] = true;
    }
}
else
{
    // Value contains symbols. <Error Message>
    ValidFields[2] = false;
    lblErrorPlaceholder4.Show();
    lblErrorPlaceholder4.Text = "please Remove Symbols or
punctuation.";
}
}
else
{
    // No Value Present. <Error Message>
    ValidFields[2] = false;
    lblErrorPlaceholder4.Show();
    lblErrorPlaceholder4.Text = "Data Required. Please Complete.";
}

// DATE OF BIRTH
if (tbFieldInfo5.Text != "")
{
    // Value is present. <Run IF Statement>

    EnteredDate = Convert.ToDateTime(tbFieldInfo5.Text);
    if (EnteredDate > DateTime.Today)
    {
        // Entered Date > CurrentDate. <Error Message>
        ValidFields[3] = false;
        lblErrorPlaceholder5.Show();
        lblErrorPlaceholder5.Text = "Please select prior date.";
    }
    else
    {
        // Date is Valid. <Validate Field>.
        ValidFields[3] = true;
    }
}
else
{
    // No Value Present. <Error Message>
    ValidFields[3] = false;
    lblErrorPlaceholder5.Show();
    lblErrorPlaceholder5.Text = "Data Required. Please use Calender
provided.";
}
}

// ADDRESS
int y = 0;
SpacePresent = false;

```

```
SpaceLocation = 0;
containsLetters = true;
ContainsNumbers = true;
FirstSection = "";
SecondSection = "";
PresenceOfPunctuation = tbFieldInfo4.Text.Any(char.IsPunctuation);
PresenceOfSymbols = tbFieldInfo4.Text.Any(char.IsSymbol);

if (tbFieldInfo6.Text != "")
{
    SpacePresent = false;
    StringCharacters = tbFieldInfo6.Text.ToCharArray();

    while (SpacePresent == false)
    {
        while (y < StringCharacters.Length)
        {
            if (StringCharacters[y] == ' ')
            {
                SpaceLocation = y;
                break;
            }
            y++;
        }
        break;
    }

    if (SpaceLocation > 0)
    {
        SpacePresent = true;
    }
}

if (tbFieldInfo6.Text != "")
{
    // If Value is present. <Run Statement>

    if (PresenceOfSymbols == false && PresenceOfPunctuation == false)
    {
        // If No symbols present. <Run Statement>

        if (SpacePresent == true)
        {
            // These lines of code segment the address number and the
            street.
            FirstSection = tbFieldInfo6.Text.Substring(0, (y));
            SecondSection = tbFieldInfo6.Text.Substring((y + 1),
            ((tbFieldInfo6.TextLength - FirstSection.Length) - 1));

            // Test address number for text.
            containsLetters = FirstSection.Any(Char.IsLetter);

            // Test the address name for numbers.
            ContainsNumbers = SecondSection.Any(Char.IsNumber);

            if (FirstSection != "" && SecondSection != "")
            {
                if (containsLetters == false && ContainsNumbers == false)
                {
```

```
        // If both sections are validated. <Validate this
field>
        ValidFields[4] = true;
    }
    else if (containsLetters == true && ContainsNumbers ==
false)
    {
        // If first section invalid, yet second valid. <Error
Message>
        ValidFields[4] = false;
        lblErrorPlaceholder6.Show();
        lblErrorPlaceholder6.Text = "Please Remove
alphametical Characters from street number.";

    }
    else if (containsLetters == false && ContainsNumbers ==
true)
    {
        // If first section valid, yet second invalid. <Error
Message>
        ValidFields[4] = false;
        lblErrorPlaceholder6.Show();
        lblErrorPlaceholder6.Text = "Please Remove Numerical
Characters from street name.";
    }
    else if (containsLetters == true && ContainsNumbers ==
true)
    {
        // If first section valid, yet second invalid. <Error
Message>
        ValidFields[4] = false;
        lblErrorPlaceholder6.Show();
        lblErrorPlaceholder6.Text = "Invalid Characters.
Please Review.";
    }
}
else
{
    // Space not present, thus field does not contain proper
format. <Error Message>
    ValidFields[4] = false;
    lblErrorPlaceholder6.Show();
    lblErrorPlaceholder6.Text = "Space Neccessary between house
Number and Street.";
}
else
{
    // Symbol present in value. <Error Message>
    ValidFields[4] = false;
    lblErrorPlaceholder6.Show();
    lblErrorPlaceholder6.Text = "Please Remove Symbol or
punctuation.";
}
else
{
    // Field does not contain text. <Error Message>
    ValidFields[4] = false;
    lblErrorPlaceholder6.Show();
    lblErrorPlaceholder6.Text = "Data Required. Please Complete.";
```

```
}

// TOWN
ContainsNumbers = tbFieldInfo7.Text.Any(char.IsDigit);
PresenceOfPunctuation = tbFieldInfo7.Text.Any(char.IsPunctuation);
PresenceOfSymbols = tbFieldInfo7.Text.Any(char.IsSymbol);

if (tbFieldInfo7.Text != "")
{
    // If Value is present within Textbox. <Run Statement>
    if (PresenceOfPunctuation == false && PresenceOfSymbols == false)
    {

        // No invalid symbols present. <Run Statement>
        if (ContainsNumbers == true)
        {
            // Value contains numbers. <Error Message>
            ValidFields[5] = false;
            lblErrorPlaceholder7.Show();
            lblErrorPlaceholder7.Text = "Please remove numerical
characters.";
        }
        else
        {
            // Value contains no invalid characters. <Validate Field>
            ValidFields[5] = true;
        }
    }
    else
    {
        // Value contains symbols. <Error Message>
        ValidFields[5] = false;
        lblErrorPlaceholder7.Show();
        lblErrorPlaceholder7.Text = "Please remove symbols and
punctuation.";
    }
}
else
{
    // No value present. <Error Message>
    ValidFields[5] = false;
    lblErrorPlaceholder7.Show();
    lblErrorPlaceholder7.Text = "Data Required. Please Complete.";
}

// POSTCODE
if (tbFieldInfo8.Text != "")
{
    if (tbFieldInfo8.TextLength == tbFieldInfo8.MaxLength)
    {
        Postcode_Value = tbFieldInfo8.Text;
        Postcode_BT = Postcode_Value.Substring(0, 2);
        Postcode_Number = Postcode_Value.Substring(2, 3);
        Postcode_Letters = Postcode_Value.Substring(5, 2);

        // Checking if 'BT' is present at start.
```

```
        if (Postcode_BT == "BT" || Postcode_BT == "bt" || Postcode_BT ==  
"Bt" || Postcode_BT == "bT")  
    {  
        // check the next three digits  
        Presence_Punctuation =  
Postcode_Number.Any(Char.IsPunctuation);  
        Presence_Letter = Postcode_Number.Any(Char.IsLetter);  
        Presence_Symbol = Postcode_Number.Any(Char.IsSymbol);  
        if (Presence_Letter == false && PresenceOfPunctuation == false  
&& Presence_Symbol == false)  
        {  
            // Check final 2 digits  
            Presence_Punctuation =  
Postcode_Letters.Any(Char.IsPunctuation);  
            Presence_Number = Postcode_Letters.Any(Char.IsNumber);  
            Presence_Symbol = Postcode_Letters.Any(Char.IsSymbol);  
            if (Presence_Number == false && Presence_Punctuation ==  
false && Presence_Symbol == false)  
            {  
                // Field valid  
                ValidFields[6] = true;  
            }  
            else  
            {  
                // Invalid characters present within letter segment  
                ValidFields[6] = false;  
                lblErrorPlaceholder8.Show();  
                lblErrorPlaceholder8.Text = "[PostCode Format =  
LL000LL] Please Replace final digits with valid characters.";  
            }  
        }  
        else  
        {  
            // Invalid value present within number segment  
            ValidFields[6] = false;  
            lblErrorPlaceholder8.Show();  
            lblErrorPlaceholder8.Text = "[PostCode Format = LL000LL]  
Please Replace '000' section with valid characters";  
        }  
    }  
    else  
    {  
        // BT not present  
        ValidFields[6] = false;  
        lblErrorPlaceholder8.Show();  
        lblErrorPlaceholder8.Text = "[PostCode Format = LL000LL] First  
characters must be 'BT'.";  
    }  
    else  
    {  
        // Box must contain 7 characters  
        ValidFields[6] = false;  
        lblErrorPlaceholder8.Show();  
        lblErrorPlaceholder8.Text = "[PostCode Format = LL000LL] Please  
use full format.";  
    }  
}  
else  
{  
    // No value present  
    ValidFields[6] = false;
```

```
        lblErrorPlaceholder8.Show();
        lblErrorPlaceholder8.Text = "Data Required. Please Complete.";
    }

    // CONTACT NUMBER
    Presence_Letter = tbFieldInfo9.Text.Any(Char.IsLetter);
    Presence_Punctuation = tbFieldInfo9.Text.Any(Char.IsPunctuation);
    Presence_Symbol = tbFieldInfo9.Text.Any(Char.IsSymbol);

    if (tbFieldInfo9.Text != "" &&
!string.IsNullOrEmpty(tbFieldInfo9.Text))
    {
        if (Presence_Letter == false)
        {
            if (Presence_Punctuation == false)
            {
                if (Presence_Symbol == false)
                {
                    ValidFields[7] = true;
                }
                else
                {
                    // <Error> Symbol Present
                    ValidFields[7] = false;
                    lblErrorPlaceholder9.Show();
                    lblErrorPlaceholder9.Text = "Please Remove symbols.";
                }
            }
            else
            {
                // <Error> Punctuation Present
                ValidFields[7] = false;
                lblErrorPlaceholder9.Show();
                lblErrorPlaceholder9.Text = "Please remove punctuation.";
            }
        }
        else
        {
            // <Error> Letter Present
            ValidFields[7] = false;
            lblErrorPlaceholder9.Show();
            lblErrorPlaceholder9.Text = "Please Remove Alphabetical
Characters.";
        }
    }
    else
    {
        ValidFields[7] = false;
        lblErrorPlaceholder9.Show();
        lblErrorPlaceholder9.Text = "Data Required. Please Complete.";
    }

    // Email Address
    if (tbFieldInfo10 != null &&
!string.IsNullOrEmpty(tbFieldInfo10.Text))
    {
        ATCount = 0;
```

```
        for (int x = 0; x < tbFieldInfo10.Text.Length; x++)
    {
        SubstringTest = tbFieldInfo10.Text.Substring(x, 1);
        if (SubstringTest == "@")
        {
            ATCount++;
        }

    }

    if (ATCount == 1)
    {
        // Valid Value
        ValidFields[8] = true;
    }
    else if (ATCount < 1)
    {
        // No @ Present <Error>
        ValidFields[8] = false;
        lblErrorPlaceholder10.Show();
        lblErrorPlaceholder10.Text = "Email Must Contain '@'..";
    }
    else if (ATCount > 1)
    {
        // No @ Present <Error>
        ValidFields[8] = false;
        lblErrorPlaceholder10.Show();
        lblErrorPlaceholder10.Text = "Only one '@' may be present within
Email.";
    }
}
else
{
    // No value Present <Error>
    ValidFields[8] = false;
    lblErrorPlaceholder10.Show();
    lblErrorPlaceholder10.Text = "Data Required. Please Complete.";
}

// GRADE ID
if (cbFieldInfo1.Text != null)
{
    if (Convert.ToInt32(cbFieldInfo1.Text) > 0 ||
Convert.ToInt32(cbFieldInfo1.Text) < Grade_ID.Count())
    {
        // Valio Value <Correct>
        ValidFields[9] = true;
    }
    else
    {
        // Invalid Value <Error>
        ValidFields[9] = false;
        lblErrorPlaceholder11.Show();
        lblErrorPlaceholder11.Text = "Entered Value does not represent a
Grade Record. Please select from provided list.";
    }
}
else
{
    // No value Present <Error>
```

```
        ValidFields[9] = false;
        lblErrorPlaceholder11.Show();
        lblErrorPlaceholder11.Text = "Data Required. Please Complete.";
    }

    // Instrument ID
    if (cbFieldInfo2.Text != null)
    {
        ValidFields[10] = true;
    }

    // Tuition Fee
    ValidFields[11] = true;
}

public void Validation_Teacher()
{
    // FIRST NAME
    if (tbFieldInfo2.Text != null &&
!string.IsNullOrEmpty(tbFieldInfo2.Text))
    {
        Presence_Number = tbFieldInfo2.Text.Any(Char.IsNumber);
        Presence_Punctuation = tbFieldInfo2.Text.Any(Char.IsPunctuation);
        Presence_Symbol = tbFieldInfo2.Text.Any(Char.IsSymbol);

        if (Presence_Number == false && Presence_Punctuation == false &&
Presence_Symbol == false)
        {
            // Valid Character Present
            ValidFields[0] = true;
        }
        else
        {
            // <Error> Invalid Characters
            ValidFields[0] = false;
            lblErrorPlaceholder2.Show();
            lblErrorPlaceholder2.Text = "Please Remove Invalid Characters.
Name should only contain Letters!";
        }
    }
    else
    {
        // <Error> No value present
        ValidFields[0] = false;
        lblErrorPlaceholder2.Show();
        lblErrorPlaceholder2.Text = "Data Required. Please Complete";
    }

    // SURNAME
    if (tbFieldInfo3.Text != null &&
!string.IsNullOrEmpty(tbFieldInfo3.Text))
    {
        Presence_Number = tbFieldInfo3.Text.Any(Char.IsNumber);
        Presence_Punctuation = tbFieldInfo3.Text.Any(Char.IsPunctuation);
        Presence_Symbol = tbFieldInfo3.Text.Any(Char.IsSymbol);

        if (Presence_Number == false && Presence_Punctuation == false &&
Presence_Symbol == false)
        {
    
```

```
// Valid Character Present
ValidFields[1] = true;
}
else
{
    // <Error> Invalid Characters
    ValidFields[1] = false;
    lblErrorPlaceholder3.Show();
    lblErrorPlaceholder3.Text = "Please Remove Invalid Characters.
Name should only contain Letters!";
}
}
else
{
    // <Error> No value present
    ValidFields[1] = false;
    lblErrorPlaceholder3.Show();
    lblErrorPlaceholder3.Text = "Data Required. Please Complete";
}

// ADDRESS
int y = 0;
SpacePresent = false;
Spacelocation = 0;
containsLetters = true;
ContainsNumbers = true;
FirstSection = "";
SecondSection = "";
PresenceOfPunctuation = tbFieldInfo4.Text.Any(char.IsPunctuation);
PresenceOfSymbols = tbFieldInfo4.Text.Any(char.IsSymbol);

if (tbFieldInfo4.Text != "")
{
    SpacePresent = false;
    StringChracters = tbFieldInfo4.Text.ToCharArray();

    while (SpacePresent == false)
    {
        while (y < StringChracters.Length)
        {
            if (StringChracters[y] == ' ')
            {
                Spacelocation = y;
                break;
            }
            y++;
        }
        break;
    }

    if (Spacelocation > 0)
    {
        SpacePresent = true;
    }
}

if (tbFieldInfo4.Text != "")
{
    // If Value is present. <Run Statement>
```

```
        if (PresenceOfSymbols == false && PresenceOfPunctuation == false)
    {
        // If No symbols present. <Run Statement>

        if (SpacePresent == true)
        {
            // These lines of code segment the address number and the
            street.

            FirstSection = tbFieldInfo4.Text.Substring(0, (y));
            SecondSection = tbFieldInfo4.Text.Substring((y + 1),
            ((tbFieldInfo4.TextLength - FirstSection.Length) - 1));

            // Test address number for text.
            containsLetters = FirstSection.Any(Char.IsLetter);

            // Test the address name for numbers.
            ContainsNumbers = SecondSection.Any(Char.IsNumber);

            if (FirstSection != "" && SecondSection != "")
            {
                if (containsLetters == false && ContainsNumbers == false)
                {
                    // If both sections are validated. <Validate this
                    field>

                    ValidFields[2] = true;
                }
                else if (containsLetters == true && ContainsNumbers ==
                false)
                {
                    // If first section invalid, yet second valid. <Error
                    Message>

                    ValidFields[2] = false;
                    lblErrorPlaceholder4.Show();
                    lblErrorPlaceholder4.Text = "Please Remove
                    alphabetical Characters from street number.";

                }
                else if (containsLetters == false && ContainsNumbers ==
                true)
                {
                    // If first section valid, yet second invalid. <Error
                    Message>

                    ValidFields[2] = false;
                    lblErrorPlaceholder4.Show();
                    lblErrorPlaceholder4.Text = "Please Remove Numerical
                    Characters from street name.";
                }
                else if (containsLetters == true && ContainsNumbers ==
                true)
                {
                    // If first section valid, yet second invalid. <Error
                    Message>

                    ValidFields[2] = false;
                    lblErrorPlaceholder4.Show();
                    lblErrorPlaceholder4.Text = "Invalid Characters.
                    Please Review.";
                }
            }
        }
    }
}
```

```

        {
            // Space not present, thus field does not contain proper
format. <Error Message>
            ValidFields[2] = false;
            lblErrorPlaceholder4.Show();
            lblErrorPlaceholder4.Text = "Space Necessary between house
Number and Street.";
        }
    }
    else
    {
        // Symbol present in value. <Error Message>
        ValidFields[2] = false;
        lblErrorPlaceholder4.Show();
        lblErrorPlaceholder4.Text = "Please Remove Symbol or
punctuation.";
    }
}
else
{
    // Field does not contain text. <Error Message>
    ValidFields[2] = false;
    lblErrorPlaceholder4.Show();
    lblErrorPlaceholder4.Text = "Data Required. Please Complete.";
}

```

```

// TOWN
ContainsNumbers = tbFieldInfo5.Text.Any(char.IsDigit);
PresenceOfPunctuation = tbFieldInfo5.Text.Any(char.IsPunctuation);
PresenceOfSymbols = tbFieldInfo5.Text.Any(char.IsSymbol);

if (tbFieldInfo5.Text != "")
{
    // If Value is present within Textbox. <Run Statement>
    if (PresenceOfPunctuation == false && PresenceOfSymbols == false)
    {

        // No invalid symbols present. <Run Statement>
        if (ContainsNumbers == true)
        {
            // Value contains numbers. <Error Message>
            ValidFields[3] = false;
            lblErrorPlaceholder5.Show();
            lblErrorPlaceholder5.Text = "Please remove numerical
characters.";
        }
    }
    else
    {
        // Value contains no invalid characters. <Validate Field>
        ValidFields[3] = true;
    }
}
else
{

```

```
// Value contains symbols. <Error Message>
ValidFields[3] = false;
lblErrorPlaceholder5.Show();
lblErrorPlaceholder5.Text = "Please remove symbols and
punctuation.";
}
}
else
{
    // No value present. <Error Message>
ValidFields[3] = false;
lblErrorPlaceholder5.Show();
lblErrorPlaceholder5.Text = "Data Required. Please Complete.";
}

// POSTCODE
if (tbFieldInfo6.Text != "")
{
    if (tbFieldInfo6.TextLength == tbFieldInfo6.MaxLength)
    {
        Postcode_Value = tbFieldInfo6.Text;
        Postcode_BT = Postcode_Value.Substring(0, 2);
        Postcode_Number = Postcode_Value.Substring(2, 3);
        Postcode_Letters = Postcode_Value.Substring(5, 2);

        // Checking if 'BT' is present at start.
        if (Postcode_BT == "BT" || Postcode_BT == "bt" || Postcode_BT ==
"Bt" || Postcode_BT == "bT")
        {
            // check the next three digits
            Presence_Punctuation =
Postcode_Number.Any(Char.IsPunctuation);
            Presence_Letter = Postcode_Number.Any(Char.IsLetter);
            Presence_Symbol = Postcode_Number.Any(Char.IsSymbol);
            if (Presence_Letter == false && PresenceOfPunctuation == false
&& Presence_Symbol == false)
            {
                // Check final 2 digits
                Presence_Punctuation =
Postcode_Letters.Any(Char.IsPunctuation);
                Presence_Number = Postcode_Letters.Any(Char.IsNumber);
                Presence_Symbol = Postcode_Letters.Any(Char.IsSymbol);
                if (Presence_Number == false && Presence_Punctuation ==
false && Presence_Symbol == false)
                {
                    // Field valid
                    ValidFields[4] = true;
                }
                else
                {
                    // Invalid characters present within letter segment
                    ValidFields[4] = false;
                    lblErrorPlaceholder6.Show();
                    lblErrorPlaceholder6.Text = "[PostCode Format =
LL000LL] Please Replace final digits with valid characters.";
                }
            }
            else
            {
                // Invalid value present within number segment
            }
        }
    }
}
```

```
        ValidFields[4] = false;
        lblErrorPlaceholder6.Show();
        lblErrorPlaceholder6.Text = "[PostCode Format = LL000LL]
Please Replace '000' section with valid characters";
    }
}
else
{
    // BT not present
    ValidFields[4] = false;
    lblErrorPlaceholder6.Show();
    lblErrorPlaceholder6.Text = "[PostCode Format = LL000LL] First
characters must be 'BT'.";
}
else
{
    // Box must contain 7 characters
    ValidFields[4] = false;
    lblErrorPlaceholder6.Show();
    lblErrorPlaceholder6.Text = "[PostCode Format = LL000LL] Please
use full format.";
}
}
else
{
    // No value present
    ValidFields[4] = false;
    lblErrorPlaceholder6.Show();
    lblErrorPlaceholder6.Text = "Data Required. Please Complete.";
}
```

```
// CONTACT NUMBER
Presence_Letter = tbFieldInfo8.Text.Any(Char.IsLetter);
Presence_Punctuation = tbFieldInfo8.Text.Any(Char.IsPunctuation);
Presence_Symbol = tbFieldInfo8.Text.Any(Char.IsSymbol);

if (tbFieldInfo8.Text != "" &&
!string.IsNullOrWhiteSpace(tbFieldInfo8.Text))
{
    if (Presence_Letter == false)
    {
        if (Presence_Punctuation == false)
        {
            if (Presence_Symbol == false)
            {
                ValidFields[5] = true;
            }
            else
            {
                // <Error> Symbol Present
                ValidFields[5] = false;
                lblErrorPlaceholder8.Show();
                lblErrorPlaceholder8.Text = "Please Remove symbols.";
            }
        }
        else
        {
```

```
        // <Error> Punctuation Present
        ValidFields[5] = false;
        lblErrorPlaceholder8.Show();
        lblErrorPlaceholder8.Text = "Please remove punctuation.";
    }
}
else
{
    // <Error> Letter Present
    ValidFields[5] = false;
    lblErrorPlaceholder8.Show();
    lblErrorPlaceholder8.Text = "Please Remove Alphabetical
Characters.";
}
}
else
{
    ValidFields[5] = false;
    lblErrorPlaceholder8.Show();
    lblErrorPlaceholder8.Text = "Data Required. Please Complete.";
}

// Email Address
if (tbFieldInfo7 != null)
{
    ATCount = 0;

    for (int x = 0; x < tbFieldInfo7.Text.Length; x++)
    {
        SubstringTest = tbFieldInfo7.Text.Substring(x, 1);
        if (SubstringTest == "@")
        {
            ATCount++;
        }
    }

    if (ATCount == 1)
    {
        // Valid Value
        ValidFields[6] = true;
    }
    else if (ATCount < 1)
    {
        // No @ Present <Error>
        ValidFields[6] = false;
        lblErrorPlaceholder7.Show();
        lblErrorPlaceholder7.Text = "Email Must Contain '@'." ;
    }
    else if (ATCount > 1)
    {
        // No @ Present <Error>
        ValidFields[6] = false;
        lblErrorPlaceholder7.Show();
        lblErrorPlaceholder7.Text = "Only one '@' may be present within
Email." ;
    }
}
```

```
        }

    else
    {
        // No value Present <Error>
        ValidFields[6] = false;
        lblErrorPlaceholder7.Show();
        lblErrorPlaceholder7.Text = "Data Required. Please Complete.";
    }

    // SPECIALISATION
    ValidFields[7] = true;

    // ROOM ID
    ValidFields[8] = true;

}

public void Validation_Instrument()
{
    // TYPE
    ValidFields[0] = true;

    // NAME
    Presence_Number = tbFieldInfo3.Text.Any(Char.IsNumber);
    Presence_Punctuation = tbFieldInfo3.Text.Any(Char.IsPunctuation);
    Presence_Symbol = tbFieldInfo3.Text.Any(Char.IsSymbol);

    if (tbFieldInfo3.Text != null &&
!string.IsNullOrWhiteSpace(tbFieldInfo3.Text))
    {
        if (Presence_Number == false)
        {
            if (Presence_Punctuation == false)
            {
                if (Presence_Symbol == false)
                {
                    ValidFields[1] = true;
                }
                else
                {
                    // <Error> Symbol Present
                    ValidFields[1] = false;
                    lblErrorPlaceholder3.Show();
                    lblErrorPlaceholder3.Text = "Please Remove Symbol
Character.";
                }
            }
            else
            {
                // <Error> Punctuation Present
                ValidFields[1] = false;
                lblErrorPlaceholder3.Show();
                lblErrorPlaceholder3.Text = "Please Remove Punctuation.";
            }
        }
        else
        {
            // <Error> Number Present
        }
    }
}
```

```
        ValidFields[1] = false;
        lblErrorPlaceholder3.Show();
        lblErrorPlaceholder3.Text = "Please Remove Numeric Character.";
    }
}
else
{
    // <Error> No value present
    ValidFields[1] = false;
    lblErrorPlaceholder3.Show();
    lblErrorPlaceholder3.Text = "Data Required. Please Complete.";
}

// QUANTITY
Presence_Letter = tbFieldInfo4.Text.Any(Char.IsLetter);
Presence_Punctuation = tbFieldInfo4.Text.Any(Char.IsPunctuation);
Presence_Symbol = tbFieldInfo4.Text.Any(Char.IsSymbol);

if (tbFieldInfo4.Text != null &&
!string.IsNullOrWhiteSpace(tbFieldInfo4.Text))
{
    if (Presence_Letter != true)
    {
        if (Presence_Punctuation != true)
        {
            if (Presence_Symbol != true)
            {

MessageBox.Show(Convert.ToInt32(tbFieldInfo4.Text).ToString());

        if (Convert.ToInt32(tbFieldInfo4.Text) > 0 &&
Convert.ToInt32(tbFieldInfo4.Text) < TotalInstruments)
        {
            // Valid Value
            ValidFields[2] = true;
        }
        else
        {
            // <Error> Value Exceeds limit
            ValidFields[2] = false;
            lblErrorPlaceholder4.Show();
            lblErrorPlaceholder4.Text = String.Format("Value
Excceds limit. Please stay within the range of (1 - {0})", TotalInstruments);
        }
    }
    else
    {
        // <Error> Symbol Present
        ValidFields[2] = false;
        lblErrorPlaceholder4.Show();
        lblErrorPlaceholder4.Text = "Please Remove Symbol
Character.";
    }
}
else
{
    // <Error> Punctuation Present
    ValidFields[2] = false;
    lblErrorPlaceholder4.Show();
    lblErrorPlaceholder4.Text = "Please Remove Puntuation.";
}
}
```

```
        }
    else
    {
        // <Error> Letter Present
        ValidFields[2] = false;
        lblErrorPlaceholder4.Show();
        lblErrorPlaceholder4.Text = "Please Remove Alphabetical
Characters.";
    }
}
else
{
    // <Error> No value present
    ValidFields[2] = false;
    lblErrorPlaceholder4.Show();
    lblErrorPlaceholder4.Text = "Data Required. Please Complete.";
}

}

public void Validation_Grade()
{
    // Grade Level (Name)
    Presence_Number = tbFieldInfo2.Text.Any(Char.IsNumber);
    Presence_Punctuation = tbFieldInfo2.Text.Any(Char.IsPunctuation);
    Presence_Symbol = tbFieldInfo2.Text.Any(Char.IsSymbol);

    if (tbFieldInfo2.Text != null &&
!string.IsNullOrWhiteSpace(tbFieldInfo2.Text))
    {
        if (Presence_Number == false)
        {
            if (Presence_Punctuation == false)
            {
                if (Presence_Symbol == false)
                {
                    // <Valid> Information
                    ValidFields[0] = true;
                }
                else
                {
                    // <Error> Symbol Present
                    ValidFields[0] = false;
                    lblErrorPlaceholder2.Show();
                    lblErrorPlaceholder2.Text = "Please Remove any Symbols.";
                }
            }
            else
            {
                // <Error> Punctuation Present
                ValidFields[0] = false;
                lblErrorPlaceholder2.Show();
                lblErrorPlaceholder2.Text = "Please Remove Punctual
characters.";
            }
        }
    }
    else
    {
        // <Error> Number Present
        ValidFields[0] = false;
        lblErrorPlaceholder2.Show();
        lblErrorPlaceholder2.Text = "Please Remove Numerical characters.";
    }
}
```

```
        }
    }
else
{
    // <Error> No value present
ValidFields[0] = false;
lblErrorPlaceholder2.Show();
lblErrorPlaceholder2.Text = "Data Required. Please Complete.";
}

// Grade Fee
Presence_Letter = tbFieldInfo3.Text.Any(Char.IsLetter);
Presence_Punctuation = tbFieldInfo3.Text.Any(Char.IsPunctuation);
Presence_Symbol = tbFieldInfo3.Text.Any(Char.IsSymbol);

if (tbFieldInfo3.Text != null &&
!string.IsNullOrWhiteSpace(tbFieldInfo3.Text))
{
    if (Presence_Letter != true)
    {
        if (Presence_Punctuation != true)
        {
            if (Presence_Symbol != true)
            {
                if (Convert.ToInt32(tbFieldInfo3.Text) > 0 &&
Convert.ToInt32(tbFieldInfo3.Text) < TotalGradeRange)
                {
                    // Valid Value
                    ValidFields[1] = true;
                }
                else
                {
                    // <Error> Value Exceeds limit
                    ValidFields[1] = false;
                    lblErrorPlaceholder3.Show();
                    lblErrorPlaceholder3.Text = String.Format("Value
Exceeds limit. Please stay within the range of (1 - {0})", TotalInstruments);
                }
            }
            else
            {
                // <Error> Symbol Present
                ValidFields[1] = false;
                lblErrorPlaceholder3.Show();
                lblErrorPlaceholder3.Text = "Please Remove Symbol
Character.";
            }
        }
        else
        {
            // <Error> Punctuation Present
            ValidFields[1] = false;
            lblErrorPlaceholder3.Show();
            lblErrorPlaceholder3.Text = "Please Remove Punctuation.";
        }
    }
    else
    {
        // <Error> Letter Present
        ValidFields[1] = false;
        lblErrorPlaceholder3.Show();
    }
}
```

```
        lblErrorPlaceholder3.Text = "Please Remove Alphabetical
Characters.";
    }
}
else
{
    // <Error> No value present
    ValidFields[1] = false;
    lblErrorPlaceholder3.Show();
    lblErrorPlaceholder3.Text = "Data Required. Please Complete.";
}

public void Validation_Room()
{
    // Room Type
    ValidFields[0] = true;

    // Specialisation
    ValidFields[1] = true;

}

public void Validation_LessonBundle()
{
    // Lesson Bundle
    Presence_Punctuation = tbFieldInfo2.Text.Any(Char.IsPunctuation);
    Presence_Symbol = tbFieldInfo2.Text.Any(Char.IsSymbol);

    if (tbFieldInfo2.Text != null &&
!string.IsNullOrWhiteSpace(tbFieldInfo2.Text))
    {
        if (Presence_Punctuation == false)
        {
            if (Presence_Symbol == false)
            {
                // <Valid> Information
                ValidFields[0] = true;
            }
            else
            {
                // <Error> Symbol Present
                ValidFields[0] = false;
                lblErrorPlaceholder2.Show();
                lblErrorPlaceholder2.Text = "Please Remove any Symbols.";
            }
        }
        else
        {
            // <Error> Punctuation Present
            ValidFields[0] = false;
            lblErrorPlaceholder2.Show();
            lblErrorPlaceholder2.Text = "Please Remove Puncutal characters.";
        }
    }
    else
    {
        // <Error> No value present
        ValidFields[0] = false;
        lblErrorPlaceholder2.Show();
    }
}
```

```
        lblErrorPlaceholder2.Text = "Data Required. Please Complete.";
    }

    // Bundle Cost
Presence_Letter = tbFieldInfo3.Text.Any(Char.IsLetter);
Presence_Punctuation = tbFieldInfo3.Text.Any(Char.IsPunctuation);
Presence_Symbol = tbFieldInfo3.Text.Any(Char.IsSymbol);

    if (tbFieldInfo3.Text != null &&
!string.IsNullOrEmpty(tbFieldInfo3.Text))
{
    if (Presence_Letter != true)
    {
        if (Presence_Punctuation != true)
        {
            if (Presence_Symbol != true)
            {
                if (Convert.ToInt32(tbFieldInfo3.Text) > 0 &&
Convert.ToInt32(tbFieldInfo3.Text) < TotalBundleRange)
                {
                    // Valid Value
                    ValidFields[1] = true;
                }
                else
                {
                    // <Error> Value Exceeds limit
                    ValidFields[1] = false;
                    lblErrorPlaceholder3.Show();
                    lblErrorPlaceholder3.Text = String.Format("Value
Exceeds limit. Please stay within the range of (1 - {0})", TotalBundleRange);
                }
            }
            else
            {
                // <Error> Symbol Present
                ValidFields[1] = false;
                lblErrorPlaceholder3.Show();
                lblErrorPlaceholder3.Text = "Please Remove Symbol
Character.";
            }
        }
    }
    else
    {
        // <Error> Punctuation Present
        ValidFields[1] = false;
        lblErrorPlaceholder3.Show();
        lblErrorPlaceholder3.Text = "Please Remove Punctuation.";
    }
}
else
{
    // <Error> Letter Present
    ValidFields[1] = false;
    lblErrorPlaceholder3.Show();
    lblErrorPlaceholder3.Text = "Please Remove Alphabetical
Characters.";
}
}
else
```

```
{  
    // <Error> No value present  
    ValidFields[1] = false;  
    lblErrorPlaceholder3.Show();  
    lblErrorPlaceholder3.Text = "Data Required. Please Complete.";  
}  
  
  
// Multiplier (Discount Rate)  
Presence_Letter = tbFieldInfo4.Text.Any(Char.IsLetter);  
Presence_Punctuation = tbFieldInfo4.Text.Any(Char.IsPunctuation);  
Presence_Symbol = tbFieldInfo4.Text.Any(Char.IsSymbol);  
  
if (tbFieldInfo4.Text != null &&  
!string.IsNullOrWhiteSpace(tbFieldInfo4.Text))  
{  
    if (Presence_Letter != true)  
    {  
        if (Presence_Punctuation != true)  
        {  
            if (Presence_Symbol != true)  
            {  
                if (Convert.ToInt32(tbFieldInfo4.Text) > 0 &&  
Convert.ToInt32(tbFieldInfo4.Text) <= 1)  
                {  
                    // Valid Value  
                    ValidFields[2] = true;  
                }  
                else  
                {  
                    // <Error> Value Exceeds limit  
                    ValidFields[2] = false;  
                    lblErrorPlaceholder4.Show();  
                    lblErrorPlaceholder4.Text = ("Value Exceeds limit.  
Please stay within the range of (0 AND 1)");  
                }  
            }  
            else  
            {  
                // <Error> Symbol Present  
                ValidFields[2] = false;  
                lblErrorPlaceholder4.Show();  
                lblErrorPlaceholder4.Text = ("Please Remove Symbol  
Character.");  
            }  
        }  
    }  
    else  
    {  
        // <Error> Punctuation Present  
        ValidFields[2] = false;  
        lblErrorPlaceholder4.Show();  
        lblErrorPlaceholder4.Text = ("Please Remove Punctuation.");  
    }  
}  
else  
{  
    // <Error> Letter Present  
    ValidFields[2] = false;  
    lblErrorPlaceholder4.Show();  
}
```

```
        lblErrorPlaceholder4.Text = ("Please Remove Alphabetical  
Characters.");  
    }  
}  
else  
{  
    // <Error> No value present  
    ValidFields[2] = false;  
    lblErrorPlaceholder4.Show();  
    lblErrorPlaceholder4.Text = ("Data Required. Please Complete.");  
}  
}  
  
public void Validation_ScheduledLesson()  
{  
    // Student ID  
    ValidFields[0] = true;  
  
    // Teacher ID  
    ValidFields[1] = true;  
  
    // Purchased Lesson ID  
    if (tbFieldInfo4.Text != null &&  
!string.IsNullOrWhiteSpace(tbFieldInfo4.Text))  
    {  
        ValidFields[2] = true;  
    }  
    else  
{  
        // <Error> No value present  
        ValidFields[2] = false;  
        lblErrorPlaceholder4.Show();  
        lblErrorPlaceholder4.Text = "Please Select Purchased Lesson. If none  
exist, then this student must first make a purchase."  
    }  
  
    // Number of Weeks  
    ValidFields[3] = true;  
  
    // Start Date  
    if (tbFieldInfo6.Text != null &&  
!string.IsNullOrWhiteSpace(tbFieldInfo6.Text))  
    {  
        ValidFields[4] = true;  
    }  
    else  
{  
        // <Error> No value present  
        ValidFields[4] = false;  
        lblErrorPlaceholder6.Show();  
        lblErrorPlaceholder6.Text = "Please select a date through the  
associated button and thus calendar."  
    }  
  
    // Booked Time  
    ValidFields[5] = true;  
  
    // End Date  
    if (tbFieldInfo9.Text != null &&  
!string.IsNullOrWhiteSpace(tbFieldInfo4.Text))  
    {  
        ValidFields[7] = true;
```

```
        }
    else
    {
        // <Error> No value present
        ValidFields[7] = false;
        lblErrorPlaceholder9.Show();
        lblErrorPlaceholder9.Text = "Please select a date through the
associated button and thus calendar.";
    }

    // Booked Day(s)
    TotalSelectedDays = 0;

    if (checkDay_Monday.CheckState == CheckState.Checked)
    {
        TotalSelectedDays++;
    }
    if (checkDay_Tuesday.CheckState == CheckState.Checked)
    {
        TotalSelectedDays++;
    }
    if (checkDay_Wednesday.CheckState == CheckState.Checked)
    {
        TotalSelectedDays++;
    }
    if (checkDay_Thursday.CheckState == CheckState.Checked)
    {
        TotalSelectedDays++;
    }
    if (checkDay_Friday.CheckState == CheckState.Checked)
    {
        TotalSelectedDays++;
    }

    if (TotalSelectedDays > 2)
    {
        ValidFields[6] = false;
        lblErrorPlaceholder8.Show();
        lblErrorPlaceholder8.Text = "Too many days selected. Max = 2";
    }
}

public void Validation_PurchasedLesson()
{
}

// HIDE ADDITIONAL COMPONENTS (OPERATION)
public void HideExcessInformation()
{
    if (lblFieldData2.Text == "[PlaceHolder]")
    {
        HidePlaceholder2();
        HidePlaceholder3();
        HidePlaceholder4();
        HidePlaceholder5();
        HidePlaceholder6();
        HidePlaceholder7();
        HidePlaceholder8();
        HidePlaceholder9();
        HidePlaceholder10();
    }
}
```

```
        HidePlaceholder11();
        HidePlaceholder12();
        HidePlaceholder13();
    }
    if (lblFieldData3.Text == "[PlaceHolder]")
    {
        HidePlaceholder3();
        HidePlaceholder4();
        HidePlaceholder5();
        HidePlaceholder6();
        HidePlaceholder7();
        HidePlaceholder8();
        HidePlaceholder9();
        HidePlaceholder10();
        HidePlaceholder11();
        HidePlaceholder12();
        HidePlaceholder13();
    }
    if (lblFieldData4.Text == "[PlaceHolder]")
    {
        HidePlaceholder4();
        HidePlaceholder5();
        HidePlaceholder6();
        HidePlaceholder7();
        HidePlaceholder8();
        HidePlaceholder9();
        HidePlaceholder10();
        HidePlaceholder11();
        HidePlaceholder12();
        HidePlaceholder13();
    }
    if (lblFieldData5.Text == "[PlaceHolder]")
    {
        HidePlaceholder5();
        HidePlaceholder6();
        HidePlaceholder7();
        HidePlaceholder8();
        HidePlaceholder9();
        HidePlaceholder10();
        HidePlaceholder11();
        HidePlaceholder12();
        HidePlaceholder13();
    }
    if (lblFieldData6.Text == "[PlaceHolder]")
    {
        HidePlaceholder6();
        HidePlaceholder7();
        HidePlaceholder8();
        HidePlaceholder9();
        HidePlaceholder10();
        HidePlaceholder11();
        HidePlaceholder12();
        HidePlaceholder13();
    }
    if (lblFieldData7.Text == "[PlaceHolder]")
    {
        HidePlaceholder7();
        HidePlaceholder8();
        HidePlaceholder9();
        HidePlaceholder10();
        HidePlaceholder11();
        HidePlaceholder12();
```

```
        HidePlaceholder13();
    }
    if (lblFieldData8.Text == "[PlaceHolder]")
    {
        HidePlaceholder8();
        HidePlaceholder9();
        HidePlaceholder10();
        HidePlaceholder11();
        HidePlaceholder12();
        HidePlaceholder13();
    }
    if (lblFieldData9.Text == "[PlaceHolder]")
    {
        HidePlaceholder9();
        HidePlaceholder10();
        HidePlaceholder11();
        HidePlaceholder12();
        HidePlaceholder13();
    }
    if (lblFieldData10.Text == "[PlaceHolder]")
    {
        HidePlaceholder10();
        HidePlaceholder11();
        HidePlaceholder12();
        HidePlaceholder13();
    }
    if (lblFieldData11.Text == "[PlaceHolder]")
    {
        HidePlaceholder11();
        HidePlaceholder12();
        HidePlaceholder13();
    }
    if (lblFieldData12.Text == "[PlaceHolder]")
    {
        HidePlaceholder12();
        HidePlaceholder13();
    }
    if (lblFieldData12.Text == "[PlaceHolder]")
    {
        HidePlaceholder13();
    }
}

public void Hide_Additional()
{
    cbFieldInfo1.Hide();
    cbFieldInfo2.Hide();
    cbFieldInfo3.Hide();
    cbFieldInfo4.Hide();
    CheckBox1.Hide();
    btnCalender.Hide();
    Calender1.Hide();
    lblFieldData14.Hide();
    tbFieldInfo14.Hide();
    btnCalendar2.Hide();
    btnDataGrid.Hide();
    btnTimeSelection.Hide();

    checkDay_Monday.Hide();
    checkDay_Tuesday.Hide();
    checkDay_Wednesday.Hide();
    checkDay_Thursday.Hide();
}
```

```
        checkDay_Friday.Hide();

        pbBackground_Day.Hide();
        pictureBox1.Hide();
    }

    public void Hide_Errors()
    {
        lblErrorPlaceholder2.Hide();
        lblErrorPlaceholder3.Hide();
        lblErrorPlaceholder4.Hide();
        lblErrorPlaceholder5.Hide();
        lblErrorPlaceholder6.Hide();
        lblErrorPlaceholder7.Hide();
        lblErrorPlaceholder8.Hide();
        lblErrorPlaceholder9.Hide();
        lblErrorPlaceholder10.Hide();
        lblErrorPlaceholder11.Hide();
        lblErrorPlaceholder12.Hide();
        lblErrorPlaceholder13.Hide();
    }

    // HIDE FUNCTIONS
    public void HidePlaceholder2()
    {
        lblFieldData2.Hide();
        tbFieldInfo2.Hide();
    }
    public void HidePlaceholder3()
    {
        lblFieldData3.Hide();
        tbFieldInfo3.Hide();
    }
    public void HidePlaceholder4()
    {
        lblFieldData4.Hide();
        tbFieldInfo4.Hide();
    }
    public void HidePlaceholder5()
    {
        lblFieldData5.Hide();
        tbFieldInfo5.Hide();
    }
    public void HidePlaceholder6()
    {
        lblFieldData6.Hide();
        tbFieldInfo6.Hide();
    }
    public void HidePlaceholder7()
    {
        lblFieldData7.Hide();
        tbFieldInfo7.Hide();
    }
    public void HidePlaceholder8()
    {
        lblFieldData8.Hide();
        tbFieldInfo8.Hide();
    }
    public void HidePlaceholder9()
    {
        lblFieldData9.Hide();
```

```
        tbFieldInfo9.Hide();
    }
    public void HidePlaceholder10()
    {
        lblFieldData10.Hide();
        tbFieldInfo10.Hide();
    }
    public void HidePlaceholder11()
    {
        lblFieldData11.Hide();
        tbFieldInfo11.Hide();
    }
    public void HidePlaceholder12()
    {
        lblFieldData12.Hide();
        tbFieldInfo12.Hide();
    }
    public void HidePlaceholder13()
    {
        lblFieldData13.Hide();
        tbFieldInfo13.Hide();
    }

    // CALENDAR DISPLAY
    private void btnCalender_Click(object sender, EventArgs e)
    {
        CalendarButton = 1;
        btnCalendar2.Enabled = false;

        btnCalender.Hide();
        Calender1.Show();
        Calender1.Location = new Point(308, 209);
    }

    // CALENDAR SELECT OUTCOME
    private void Calender1_DateSelected(object sender, DateRangeEventArgs e)
    {
        SelectedDate = Calender1.SelectionRange.Start.ToShortDateString();

        if (GlobalVariables.PreviousForm == "StudentTable")
        {
            tbFieldInfo5.Text = SelectedDate;
            btnCalender.Show();
        }
        else if (GlobalVariables.PreviousForm == "ScheduledLessonTable" ||
GlobalVariables.PreviousForm == "Calender")
        {
            if (CalendarButton == 1)
            {
                tbFieldInfo6.Text = SelectedDate;
                btnCalender.Show();
            }
            else if (CalendarButton == 2)
            {
                tbFieldInfo9.Text = SelectedDate;
                btnCalendar2.Show();
            }
        }
    }

    btnCalender.Enabled = true;
```

```
        btnCalendar2.Enabled = true;
        Calender1.Hide();
    }

    // STORE TABLE INFORMATION LOCALLY
    public void LocalInformationStorage()
    {
        // This fuunction will be used to store various peices of information from
        the schedule Table. This will be used to complete the schedule
        using (connection = new SqlConnection(connectionString))
        {
            connection.Open();
            using (SqlCommand Command = new SqlCommand("SELECT * FROM
Scheduled_Lessons", connection))
            {

                DataReader = Command.ExecuteReader();
                while (DataReader.Read())
                {
                    // For each record, store the corrisponding peice of
                    information.

                    AllScheduledLessonsIDs.Add(DataReader.GetInt32(0));
                    AllScheduledLessonsStudentIDs.Add(DataReader.GetInt32(1));
                    AllScheduledLessonsTeacherIDs.Add(DataReader.GetInt32(2));
                    AllScheduledLessonsBundleIDs.Add(DataReader.GetInt32(3));
                    AllScheduledBookedDays.Add(DataReader.GetString(6));
                    AllScheduledLessonsStartDates.Add(DataReader.GetDateTime(5));
                    AllScheduledLessonsEndDates.Add(DataReader.GetDateTime(8));
                }
            }

            using (connection = new SqlConnection(connectionString))
            {
                connection.Open();
                using (SqlCommand Command = new SqlCommand("SELECT ScheduleID FROM
LessonDates", connection))
                {

                    DataReader = Command.ExecuteReader();
                    while (DataReader.Read())
                    {
                        // For each record, store the corrisponding peice of
                        information.

                        AllLessonDatesScheduleIDs.Add(DataReader.GetInt32(0));
                    }
                }
                DataReader.Close();
            }

            using (connection = new SqlConnection(connectionString))
            {
                using (SqlCommand cmd = new SqlCommand("SELECT * FROM
LessonDate_Archive", connection))
                {
                    connection.Open();

                    DataReader = cmd.ExecuteReader();
                    while (DataReader.Read())
                    {
                        LessonDates_Archive_ID.Add(DataReader.GetInt32(2));
                    }
                }
            }
        }
    }
}
```

```
        }

    }

    // STORE INFORMATION FOR TOTAL COST CALCULATION
    public void StoreInformation_BundleCalculator()
    {
        using (connection = new SqlConnection(connectionString))
        {
            connection.Open();
            using (SqlCommand Command4 = new SqlCommand("SELECT
LessonsPurchasedID, StudentID, LessonBundleID FROM LessonsPurchased", connection))
            {

                DataReader = Command4.ExecuteReader();
                while (DataReader.Read())
                {
                    PurchasedLessons_LessonBundleID.Add(DataReader.GetInt32(2));
                    PurchasedLessons_StudentID.Add(DataReader.GetInt32(1));
                    PurchasedLesson_IDs.Add(DataReader.GetInt32(0));

                }
            }
            using (connection = new SqlConnection(connectionString))
            {
                connection.Open();
                using (SqlCommand Command5 = new SqlCommand("SELECT * FROM
LessonBundles", connection))
                {

                    DataReader = Command5.ExecuteReader();
                    while (DataReader.Read())
                    {
                        LessonBundle_ID.Add(DataReader.GetInt32(0));
                        LessonBundle_BundleCost.Add(DataReader.GetInt32(2));
                        LessonBundle_Multiplier.Add(DataReader.GetDouble(3));
                    }
                }
            }
            using (connection = new SqlConnection(connectionString))
            {
                connection.Open();
                using (SqlCommand Command5 = new SqlCommand("SELECT StudentID, GradeID
FROM Students", connection))
                {

                    DataReader = Command5.ExecuteReader();
                    while (DataReader.Read())
                    {
                        Student_ID.Add(DataReader.GetInt32(0));
                        Student_GradeID.Add(DataReader.GetInt32(1));
                    }
                }
            }
            using (connection = new SqlConnection(connectionString))
            {
                connection.Open();
                using (SqlCommand Command5 = new SqlCommand("SELECT * FROM Grade",
connection))
                {

```

```
        DataReader = Command5.ExecuteReader();
        while (DataReader.Read())
        {
            Grade_ID.Add(DataReader.GetInt32(0));
            Grade_Cost.Add(DataReader.GetInt32(2));
        }
    }

// CALCULATE TOTAL BUNDLE COSTS
public void BundleCostCalculator()
{
    try
    {
        // Selecting each purchased lesson record individually.
        for (int x = 0; x < PurchasedLesson_IDs.Count(); x++)
        {
            // Storing the information
            Desired_PurchasedLesson_ID = PurchasedLesson_IDs[x];
            Desired_PurchasedLesson_StudentID = PurchasedLessons_StudentID[x];
            Desired_PurchasedLessons_LessonBundleID =
PurchasedLessons_LessonBundleID[x];

            // Locating the associated student.
            for (int y = 0; y < Student_ID.Count(); y++)
            {
                // Storing the associated grade ID
                if (Desired_PurchasedLesson_StudentID == Student_ID[y])
                {
                    Desired_Student_Grade = Student_GradeID[y];
                    break;
                }
            }

            // Locate the associated grade information
            for (int r = 0; r < Grade_ID.Count(); r++)
            {
                if (Desired_Student_Grade == Grade_ID[r])
                {
                    Desired_Grade_GradeFee = Grade_Cost[r];
                    break;
                }
            }

            // Locate the associated Lesson Bundle information
            for (int z = 0; z < LessonBundle_ID.Count(); z++)
            {
                if (Desired_PurchasedLessons_LessonBundleID ==
LessonBundle_ID[z])
                {
                    Desired_LessonBundle_Cost = LessonBundle_BundleCost[z];
                    Desired_LessonBundle_Multiplier =
LessonBundle_Multiplier[z];
                    break;
                }
            }

            // This will store all total cost values to a new list.
        }
    }
}
```

```
        TotalBundleCost.Add((Desired_Grade_GradeFee *  
Desired_LessonBundle_Cost) * Desired_LessonBundle_Multiplier);  
    }  
}  
catch (Exception ex)  
{  
    MessageBox.Show("Fail");  
}  
  
// UPDATE PURCHASEDLESSONS TABLE  
for (int x = 0; x < TotalBundleCost.Count(); x++)  
{  
    // Running through the list, update all records accordingly.  
    using (connection = new SqlConnection(connectionString))  
    {  
        connection.Open();  
        using (SqlCommand cmd = new SqlCommand("UPDATE LessonsPurchased  
SET Total_Bundle_Cost = @BundleCost WHERE LessonsPurchasedID = @PurchaseID",  
connection))  
        {  
            cmd.Parameters.AddWithValue("@BundleCost",  
TotalBundleCost[x]);  
            cmd.Parameters.AddWithValue("@PurchaseID",  
PurchasedLesson_IDs[x]);  
  
            // Run query  
            cmd.ExecuteNonQuery();  
        }  
    }  
}  
  
// IDENTIFY SCHEDULED DAYS  
public void ScheduledDays(string BookedDays)  
{  
    // Reset these values for new record.  
    for (int I = 0; I < WeekDays.Count(); I++)  
    {  
        WeekDays[I] = false;  
    }  
    NumberOfBookedDays = 0;  
  
    // This will identify booked day(s).  
  
    if (BookedDays.Contains("Monday"))  
    {  
        WeekDays[0] = true;  
        NumberOfBookedDays += 1;  
    }  
    if (BookedDays.Contains("Tuesday"))  
    {  
        WeekDays[1] = true;  
        NumberOfBookedDays += 1;  
    }  
    if (BookedDays.Contains("Wednesday"))  
    {  
        WeekDays[2] = true;  
        NumberOfBookedDays += 1;  
    }  
    if (BookedDays.Contains("Thursday"))  
    {  
        WeekDays[3] = true;  
        NumberOfBookedDays += 1;  
    }  
}
```

```
{  
    WeekDays[3] = true;  
    NumberOfBookedDays += 1;  
}  
if (BookedDays.Contains("Friday"))  
{  
    WeekDays[4] = true;  
    NumberOfBookedDays += 1;  
}  
}  
  
// COMPARE SCHEDULE DAYS TO UPCOMING DATES  
public void Day_Date_Comparision(string SelectedDay, DateTime ComparedDate,  
DateTime Endate)  
{  
    string DayofWeek = "";  
    bool found = false;  
  
    while (ComparedDate < Endate)  
    {  
        DayofWeek = ComparedDate.DayOfWeek.ToString();  
  
        if (DayofWeek != SelectedDay && found == false)  
        {  
            ComparedDate = ComparedDate.AddDays(1);  
        }  
        else  
        {  
            found = true;  
            AppendDate = ComparedDate;  
            AppendScheduleInformation();  
            ComparedDate = ComparedDate.AddDays(7);  
        }  
    }  
}  
  
// APPEND DATE INFORMATION INTO LESSON DATES  
public void AppendScheduleInformation()  
{  
    using (connection = new SqlConnection(connectionString))  
    {  
        connection.Open();  
        using (SqlCommand Command = new SqlCommand("INSERT INTO  
LessonDates(ScheduleID, Date, Attended) VALUES(@Value1, @Value2, @Value3)",  
connection))  
        {  
            Command.Parameters.AddWithValue("@Value1", AppendID);  
            Command.Parameters.AddWithValue("@Value2", AppendDate);  
            Command.Parameters.AddWithValue("@Value3", AppendAttend);  
            Command.ExecuteNonQuery();  
        }  
    }  
}  
  
private void btnCalendar2_Click(object sender, EventArgs e)  
{  
    CalendarButton = 2;  
    btnCalender.Enabled = false;
```

```
        btnCalendar2.Hide();
        Calender1.Show();
        Calender1.Location = new Point(314, 417);
    }

    private void btnTimeSelection_Click(object sender, EventArgs e)
    {
        checkDay_Monday.Show();
        checkDay_Tuesday.Show();
        checkDay_Wednesday.Show();
        checkDay_Thursday.Show();
        checkDay_Friday.Show();
        pbBackground_Day.Show();
        pictureBox1.Show();

        btnTimeSelection.Hide();
    }

    private void frmAddField_MouseEnter(object sender, EventArgs e)
    {
        checkDay_Monday.Hide();
        checkDay_Tuesday.Hide();
        checkDay_Wednesday.Hide();
        checkDay_Thursday.Hide();
        checkDay_Friday.Hide();
        pbBackground_Day.Hide();
        pictureBox1.Hide();
        DataGrid_PurchasedLessons.Hide();

        Calender1.Hide();

        if (GlobalVariables.PreviousForm == "ScheduledLessonTable" ||
            GlobalVariables.PreviousForm == "Calender")
        {

            btnCalender.Show();
            btnDataGrid.Show();
            btnTimeSelection.Show();

            if (GlobalVariables.Purpose == "Update")
            {
                btnCalendar2.Show();
            }
        }
        else if (GlobalVariables.PreviousForm == "StudentTable")
        {
            btnCalender.Show();
        }
    }

    private void btnDataGrid_Click(object sender, EventArgs e)
    {
        DataGrid_PurchasedLessons.Show();
        DataGrid_PurchasedLessons.Location = new Point(308, 207);
        DataGrid_PurchasedLessons.Size = new Size(444, 109);
    }

    private void cbFieldInfo1_SelectedIndexChanged(object sender, EventArgs e)
    {
        if (GlobalVariables.PreviousForm == "ScheduledLessonTable" ||
            GlobalVariables.PreviousForm == "Calender" && GlobalVariables.Purpose == "Update")
```

```
        {
            Student_Search_Breakdown();
            tbFieldInfo4.Text = null;
        }
    }

    public void Student_Search_Breakdown()
    {
        string Sub;
        for (int x = 0; x < cbFieldInfo1.Text.Length; x++)
        {
            Sub = cbFieldInfo1.Text.Substring(x, 1);
            if (Sub == " ")
            {
                Desired_StudentID = Convert.ToInt32(cbFieldInfo1.Text.Substring(0,
x));
                break;
            }
        }

        Store_PurchaseInformation();
    }

    private void DataGrid_PurchasedLessons_SelectionChanged(object sender,
EventArgs e)
{
    if (DataGrid_PurchasedLessons.SelectedCells.Count > 0)
    {
        int selectedrowindex =
DataGrid_PurchasedLessons.SelectedCells[0].RowIndex;

        DataGridViewRow selectedRow =
DataGrid_PurchasedLessons.Rows[selectedrowindex];

        string a =
Convert.ToString(selectedRow.Cells["LessonsPurchasedID"].Value);

        tbFieldInfo4.Text = a;
        DataGrid_PurchasedLessons.Hide();
        btnDataGrid.Show();
    }
}

private void DataGrid_PurchasedLessons_Click(object sender, EventArgs e)
{
    if (DataGrid_PurchasedLessons.SelectedCells.Count > 0)
    {
        int selectedrowindex =
DataGrid_PurchasedLessons.SelectedCells[0].RowIndex;

        DataGridViewRow selectedRow =
DataGrid_PurchasedLessons.Rows[selectedrowindex];

        string a =
Convert.ToString(selectedRow.Cells["LessonsPurchasedID"].Value);

        tbFieldInfo4.Text = a;
        DataGrid_PurchasedLessons.Hide();
        btnDataGrid.Show();
    }
}
```

```
    }  
}
```

FrmCalendarDates

```
using System;  
using System.Collections.Generic;  
using System.ComponentModel;  
using System.Data;  
using System.Drawing;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
using System.Windows.Forms;  
  
namespace frmSplash  
{  
    public partial class frmCalenderDates : Form  
    {  
        string SelectedDate;  
        DateTime SearchDate;  
        DateTime CurrentDate;  
        int CurrentYear;  
        string DatePlaceholder;  
        string CurrentDatePlaceholder;  
  
        public frmCalenderDates()  
        {  
            InitializeComponent();  
        }  
  
        private void frmCalenderDates_Load(object sender, EventArgs e)  
        {  
            CurrentDate = DateTime.Today;  
            CurrentDatePlaceholder = Convert.ToString(CurrentDate);  
            CurrentYear = Convert.ToInt32(CurrentDatePlaceholder.Substring(6, 4));  
        }  
  
        private void monthCalendar1_DateChanged(object sender, DateRangeEventArgs e)  
        {  
            // This block of code is used to individually store data from the selected  
            date variable.  
            SelectedDate = ScheduleCalendar.SelectionRange.Start.ToShortDateString();  
            // This identifies the selected date and assigns it to the selecteddate variable.  
            DatePlaceholder = Convert.ToString(SelectedDate); // This also converts  
            the date to a string for simple use.  
  
            // Then with the use of substrings, it is possible to individually assign  
            values to global variables.  
            GlobalVariables.SelectedDay = Convert.ToInt32(DatePlaceholder.Substring(0,  
2));  
            GlobalVariables.SelectedMonthInt =  
Convert.ToInt32(DatePlaceholder.Substring(3, 2));  
            GlobalVariables.SelectedYear =  
Convert.ToInt32(DatePlaceholder.Substring(6, 4));  
    }
```

```
}

private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    // This will provide the user with the ability to quickly access any
month.
    if (comboBox1.Text == "January")
    {
        SearchDate = Convert.ToDateTime(String.Format("01/01/{0}",
CurrentYear));
        ScheduleCalendar.SetDate(SearchDate);
    }
    else if (comboBox1.Text == "February")
    {
        SearchDate = Convert.ToDateTime(String.Format("01/02/{0}",
CurrentYear));
        ScheduleCalendar.SetDate(SearchDate);
    }
    else if (comboBox1.Text == "March")
    {
        SearchDate = Convert.ToDateTime(String.Format("01/03/{0}",
CurrentYear));
        ScheduleCalendar.SetDate(SearchDate);
    }
    else if (comboBox1.Text == "April")
    {
        SearchDate = Convert.ToDateTime(String.Format("01/04/{0}",
CurrentYear));
        ScheduleCalendar.SetDate(SearchDate);
    }
    else if (comboBox1.Text == "May")
    {
        SearchDate = Convert.ToDateTime(String.Format("01/05/{0}",
CurrentYear));
        ScheduleCalendar.SetDate(SearchDate);
    }
    else if (comboBox1.Text == "June")
    {
        SearchDate = Convert.ToDateTime(String.Format("01/06/{0}",
CurrentYear));
        ScheduleCalendar.SetDate(SearchDate);
    }
    else if (comboBox1.Text == "July")
    {
        SearchDate = Convert.ToDateTime(String.Format("01/07/{0}",
CurrentYear));
        ScheduleCalendar.SetDate(SearchDate);
    }
    else if (comboBox1.Text == "August")
    {
        SearchDate = Convert.ToDateTime(String.Format("01/08/{0}",
CurrentYear));
        ScheduleCalendar.SetDate(SearchDate);
    }
    else if (comboBox1.Text == "September")
    {
        SearchDate = Convert.ToDateTime(String.Format("01/09/{0}",
CurrentYear));
        ScheduleCalendar.SetDate(SearchDate);
    }
    else if (comboBox1.Text == "October")
```

```
{  
    SearchDate = Convert.ToDateTime(String.Format("01/10/{0}",  
CurrentYear));  
    ScheduleCalendar.SetDate(SearchDate);  
}  
else if (comboBox1.Text == "November")  
{  
    SearchDate = Convert.ToDateTime(String.Format("01/11/{0}",  
CurrentYear));  
    ScheduleCalendar.SetDate(SearchDate);  
}  
else if (comboBox1.Text == "December")  
{  
    SearchDate = Convert.ToDateTime(String.Format("01/12/{0}",  
CurrentYear));  
    ScheduleCalendar.SetDate(SearchDate);  
}  
  
}  
  
private void ScheduleCalendar_DateSelected(object sender, DateRangeEventArgs  
e)  
{  
    DateTime SelectedDate = new DateTime(GlobalVariables.SelectedYear,  
GlobalVariables.SelectedMonthInt, GlobalVariables.SelectedDay);  
    GlobalVariables.DayName = SelectedDate.ToString("dddd");  
  
    // This if statement will provide an error message should July or August  
be selected.  
    if (GlobalVariables.SelectedMonthInt == 07 ||  
GlobalVariables.SelectedMonthInt == 08)  
    {  
        MessageBox.Show("Private classes are unavailable during Summer  
Months");  
    }  
    // This statement will provide an error should a weekend be selected.  
    else if (GlobalVariables.DayName == "Saturday" || GlobalVariables.DayName  
== "Sunday")  
    {  
        MessageBox.Show("Weekends are reserved for specialised classes");  
    }  
    // Otherwise the selected month will be saved, and the next form will  
load.  
    else  
    {  
        if (GlobalVariables.SelectedMonthInt == 09)  
        {  
            GlobalVariables.SelectedMonthString = "September";  
  
            LoadCalenderTimes();  
        }  
  
        if (GlobalVariables.SelectedMonthInt == 10)  
        {  
            GlobalVariables.SelectedMonthString = "October";  
  
            LoadCalenderTimes();  
        }  
  
        if (GlobalVariables.SelectedMonthInt == 11)  
        {  
            GlobalVariables.SelectedMonthString = "November";  
        }  
    }  
}
```

```
        LoadCalenderTimes();
    }

    if (GlobalVariables.SelectedMonthInt == 12)
    {
        GlobalVariables.SelectedMonthString = "December";

        LoadCalenderTimes();
    }

    if (GlobalVariables.SelectedMonthInt == 01)
    {
        GlobalVariables.SelectedMonthString = "January";

        LoadCalenderTimes();
    }

    if (GlobalVariables.SelectedMonthInt == 02)
    {
        GlobalVariables.SelectedMonthString = "Februray";

        LoadCalenderTimes();
    }

    if (GlobalVariables.SelectedMonthInt == 03)
    {
        GlobalVariables.SelectedMonthString = "March";

        LoadCalenderTimes();
    }

    if (GlobalVariables.SelectedMonthInt == 04)
    {
        GlobalVariables.SelectedMonthString = "April";

        LoadCalenderTimes();
    }

    if (GlobalVariables.SelectedMonthInt == 05)
    {
        GlobalVariables.SelectedMonthString = "May";

        LoadCalenderTimes();
    }

    if (GlobalVariables.SelectedMonthInt == 06)
    {
        GlobalVariables.SelectedMonthString = "June";

        LoadCalenderTimes();
    }
}

private void btnReturn_Click(object sender, EventArgs e)
{
    frmPrivateTuition PrivateTuition = new frmPrivateTuition();
    PrivateTuition.Show();
    this.Hide();
}
```

```
}

private void LoadCalenderTimes()
{
    frmCalenderTimes CalenderTimes = new frmCalenderTimes();
    CalenderTimes.Show();
    this.Hide();
}

// FEEDBACK
// Return to Menu button
private void btnReturn_MouseEnter(object sender, EventArgs e)
{
    btnReturn.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}
private void btnReturn_MouseLeave(object sender, EventArgs e)
{
    btnReturn.BackColor = Color.Firebrick;
    this.Cursor = Cursors.Arrow;
}

// Calendar
private void ScheduleCalendar_MouseEnter(object sender, EventArgs e)
{
    this.Cursor = Cursors.Hand;
}
private void ScheduleCalendar_MouseLeave(object sender, EventArgs e)
{
    this.Cursor = Cursors.Arrow;
}
}
```

FrmCalendar Times

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace frmSplash
{
    public partial class frmCalenderTimes : Form
    {

        public frmCalenderTimes()
        {
            InitializeComponent();
        }

        private void frmCalenderTimes_Load(object sender, EventArgs e)
        {
            GlobalVariables.scheduleErrorMessage = true;

            // This block of code should automatically update accordingly as to
            // display the selected Date.
            GlobalVariables.SelectedScheduleDate = String.Format("{0}/{1}/{2}",
GlobalVariables.SelectedDay, GlobalVariables.SelectedMonthInt,
GlobalVariables.SelectedYear); // Establishes string format + Enters selected data in
global string.
            label1.Text = GlobalVariables.SelectedScheduleDate; // This then writes
the date string to a label present within the form.

            // This block of code will check the selected date, and update the format
accordingly for visual astetics.
            if (GlobalVariables.SelectedDay == 1 || GlobalVariables.SelectedDay == 21
|| GlobalVariables.SelectedDay == 31)
            {
                label1.Text = String.Format("{0} {1}st",
GlobalVariables.SelectedMonthString, GlobalVariables.SelectedDay);
            }
            else if (GlobalVariables.SelectedDay == 2 || GlobalVariables.SelectedDay
== 22)
            {
                label1.Text = String.Format("{0} {1}nd",
GlobalVariables.SelectedMonthString, GlobalVariables.SelectedDay);
            }
            else if (GlobalVariables.SelectedDay == 3 || GlobalVariables.SelectedDay
== 23)
            {
                label1.Text = String.Format("{0} {1}rd",
GlobalVariables.SelectedMonthString, GlobalVariables.SelectedDay);
            }
            else
            {
                label1.Text = String.Format("{0} {1}th",
GlobalVariables.SelectedMonthString, GlobalVariables.SelectedDay);
            }
        }
    }
}
```

```
        if (GlobalVariables.SelectedDay == 1 || GlobalVariables.SelectedDay == 2
|| GlobalVariables.SelectedDay == 3 || GlobalVariables.SelectedDay == 4 ||
GlobalVariables.SelectedDay == 5 || GlobalVariables.SelectedDay == 6
|| GlobalVariables.SelectedDay == 7 || GlobalVariables.SelectedDay == 8 ||
GlobalVariables.SelectedDay == 9)
{
    if (GlobalVariables.SelectedMonthInt != 12 ||
GlobalVariables.SelectedMonthInt != 11 || GlobalVariables.SelectedMonthInt != 10)
    {
        GlobalVariables.SelectedScheduleDate =
String.Format("{0}{1}/{2}", GlobalVariables.SelectedDay,
GlobalVariables.SelectedMonthInt, GlobalVariables.SelectedYear);
    }
}

// Selecting one of the buttons below will set the selectedtime variable, and
also load the next form.
private void button1_Click(object sender, EventArgs e)
{
    GlobalVariables.SelectedTime = "13:00:00";
    LoadClasSchedule(); // This code is used to run the LoadClassSchedule
Function.
}

private void button2_Click(object sender, EventArgs e)
{
    GlobalVariables.SelectedTime = "13:30:00";
    GlobalVariables.SelectedTimeTextFormat = "13:30 PM";
    LoadClasSchedule();
}

private void button3_Click(object sender, EventArgs e)
{
    GlobalVariables.SelectedTime = "14:00:00";
    GlobalVariables.SelectedTimeTextFormat = "14:00 PM";
    LoadClasSchedule();
}

private void button4_Click(object sender, EventArgs e)
{
    GlobalVariables.SelectedTime = "14:30:00";
    GlobalVariables.SelectedTimeTextFormat = "14:30 PM";
    LoadClasSchedule();
}

private void button5_Click(object sender, EventArgs e)
{
    GlobalVariables.SelectedTime = "15:00:00";
    LoadClasSchedule();
}

private void button6_Click(object sender, EventArgs e)
{
    GlobalVariables.SelectedTime = "15:30:00";
    LoadClasSchedule();
}

private void button7_Click(object sender, EventArgs e)
```

```
{  
    GlobalVariables.SelectedTime = "16:00:00";  
    LoadClasSchedule();  
}  
  
private void button8_Click(object sender, EventArgs e)  
{  
    GlobalVariables.SelectedTime = "16:30:00";  
    LoadClasSchedule();  
}  
  
private void button9_Click(object sender, EventArgs e)  
{  
    GlobalVariables.SelectedTime = "17:00:00";  
    LoadClasSchedule();  
}  
  
private void button10_Click(object sender, EventArgs e)  
{  
    GlobalVariables.SelectedTime = "17:30:00";  
    LoadClasSchedule();  
}  
  
private void button11_Click(object sender, EventArgs e)  
{  
    GlobalVariables.SelectedTime = "18:00:00";  
    LoadClasSchedule();  
}  
  
private void button12_Click(object sender, EventArgs e)  
{  
    GlobalVariables.SelectedTime = "18:30:00";  
    LoadClasSchedule();  
}  
  
private void button13_Click(object sender, EventArgs e)  
{  
    GlobalVariables.SelectedTime = "19:00:00";  
    LoadClasSchedule();  
}  
  
private void button14_Click(object sender, EventArgs e)  
{  
    GlobalVariables.SelectedTime = "19:30:00";  
    LoadClasSchedule();  
}  
  
private void button15_Click(object sender, EventArgs e)  
{  
    GlobalVariables.SelectedTime = "20:00:00";  
    LoadClasSchedule();  
}  
  
private void button16_Click(object sender, EventArgs e)  
{  
    GlobalVariables.SelectedTime = "20:30:00";  
    LoadClasSchedule();  
}  
  
private void btnReturn_Click(object sender, EventArgs e)
```

```
{  
    // Resets all variables.  
    GlobalVariables.Date1 = false;  
    GlobalVariables.Date2 = false;  
    GlobalVariables.Date3 = false;  
    GlobalVariables.Date4 = false;  
    GlobalVariables.Date5 = false;  
    GlobalVariables.Date6 = false;  
    GlobalVariables.Date7 = false;  
    GlobalVariables.Date8 = false;  
    GlobalVariables.Date9 = false;  
    GlobalVariables.Date10 = false;  
    GlobalVariables.Date11 = false;  
    GlobalVariables.Date12 = false;  
    GlobalVariables.Date13 = false;  
    GlobalVariables.Date14 = false;  
    GlobalVariables.Date15 = false;  
    GlobalVariables.Date16 = false;  
    GlobalVariables.Date17 = false;  
    GlobalVariables.Date18 = false;  
    GlobalVariables.Date19 = false;  
    GlobalVariables.Date20 = false;  
    GlobalVariables.Date21 = false;  
    GlobalVariables.Date22 = false;  
    GlobalVariables.Date23 = false;  
    GlobalVariables.Date24 = false;  
    GlobalVariables.Date25 = false;  
    GlobalVariables.Date26 = false;  
    GlobalVariables.Date27 = false;  
    GlobalVariables.Date28 = false;  
    GlobalVariables.Date29 = false;  
    GlobalVariables.Date30 = false;  
    GlobalVariables.Date31 = false;  
  
    GlobalVariables.September = false;  
    GlobalVariables.January = false;  
    GlobalVariables.February = false;  
    GlobalVariables.March = false;  
    GlobalVariables.April = false;  
    GlobalVariables.May = false;  
    GlobalVariables.June = false;  
    GlobalVariables.July = false;  
    GlobalVariables.August = false;  
    GlobalVariables.October = false;  
    GlobalVariables.November = false;  
    GlobalVariables.December = false;  
  
    // Loads previous form.  
    frmCalenderDates CalenderDates = new frmCalenderDates();  
    CalenderDates.Show();  
    this.Hide();  
}  
  
public void LoadClasSchedule()  
{  
    GlobalVariables.SelectedTimeTextFormat =  
    GlobalVariables.SelectedTime.Substring(0, 5) + " " + "PM";  
  
    // Simplu loads Class Schedule form.  
    frmClassSchedule ClassSchedule = new frmClassSchedule();  
    ClassSchedule.Show();  
    this.Hide();
```

```
}

// BUTTON FEEDBACK
// 13:00
private void button1_MouseEnter(object sender, EventArgs e)
{
    button1.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}
private void button1_MouseLeave(object sender, EventArgs e)
{
    button1.BackColor = Color.LightCyan;
    this.Cursor = Cursors.Arrow;
}

// 13:30
private void button2_MouseEnter(object sender, EventArgs e)
{
    button2.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}
private void button2_MouseLeave(object sender, EventArgs e)
{
    button2.BackColor = Color.LightCyan;
    this.Cursor = Cursors.Arrow;
}

// 14:00
private void button3_MouseEnter(object sender, EventArgs e)
{
    button3.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}
private void button3_MouseLeave(object sender, EventArgs e)
{
    button3.BackColor = Color.LightCyan;
    this.Cursor = Cursors.Arrow;
}

// 14:30
private void button4_MouseEnter(object sender, EventArgs e)
{
    button4.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}
private void button4_MouseLeave(object sender, EventArgs e)
{
    button4.BackColor = Color.LightCyan;
    this.Cursor = Cursors.Arrow;
}

// 15:00
private void button5_MouseEnter(object sender, EventArgs e)
{
    button5.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}
private void button5_MouseLeave(object sender, EventArgs e)
{
    button5.BackColor = Color.LightCyan;
    this.Cursor = Cursors.Arrow;
}
```

```
// 15:30
private void button6_MouseEnter(object sender, EventArgs e)
{
    button6.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}
private void button6_MouseLeave(object sender, EventArgs e)
{
    button6.BackColor = Color.LightCyan;
    this.Cursor = Cursors.Arrow;
}

// 16:00
private void button7_MouseEnter(object sender, EventArgs e)
{
    button7.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}
private void button7_MouseLeave(object sender, EventArgs e)
{
    button7.BackColor = Color.LightCyan;
    this.Cursor = Cursors.Arrow;
}

// 16:30
private void button8_MouseEnter(object sender, EventArgs e)
{
    button8.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}
private void button8_MouseLeave(object sender, EventArgs e)
{
    button8.BackColor = Color.LightCyan;
    this.Cursor = Cursors.Arrow;
}

// 17:00
private void button9_MouseEnter(object sender, EventArgs e)
{
    button9.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}
private void button9_MouseLeave(object sender, EventArgs e)
{
    button9.BackColor = Color.LightCyan;
    this.Cursor = Cursors.Arrow;
}

// 17:30
private void button10_MouseEnter(object sender, EventArgs e)
{
    button10.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}
private void button10_MouseLeave(object sender, EventArgs e)
{
    button10.BackColor = Color.LightCyan;
    this.Cursor = Cursors.Arrow;
}

// 18:00
```

```
private void button11_MouseEnter(object sender, EventArgs e)
{
    button11.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}
private void button11_MouseLeave(object sender, EventArgs e)
{
    button11.BackColor = Color.LightCyan;
    this.Cursor = Cursors.Arrow;
}

// 18:30
private void button12_MouseEnter(object sender, EventArgs e)
{
    button12.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}
private void button12_MouseLeave(object sender, EventArgs e)
{
    button12.BackColor = Color.LightCyan;
    this.Cursor = Cursors.Arrow;
}

// 19:00
private void button13_MouseEnter(object sender, EventArgs e)
{
    button13.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}
private void button13_MouseLeave(object sender, EventArgs e)
{
    button13.BackColor = Color.LightCyan;
    this.Cursor = Cursors.Arrow;
}

// 19:30
private void button14_MouseEnter(object sender, EventArgs e)
{
    button14.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}
private void button14_MouseLeave(object sender, EventArgs e)
{
    button14.BackColor = Color.LightCyan;
    this.Cursor = Cursors.Arrow;
}

// 20:00
private void button15_MouseEnter(object sender, EventArgs e)
{
    button15.BackColor = Color.LightSlateGray;
    this.Cursor = Cursors.Hand;
}
private void button15_MouseLeave(object sender, EventArgs e)
{
    button15.BackColor = Color.LightCyan;
    this.Cursor = Cursors.Arrow;
}

// 20:30
private void button16_MouseEnter(object sender, EventArgs e)
{
```

```
        button16.BackColor = Color.LightSlateGray;
        this.Cursor = Cursors.Hand;
    }
    private void button16_MouseLeave(object sender, EventArgs e)
    {
        button16.BackColor = Color.LightCyan;
        this.Cursor = Cursors.Arrow;
    }
}
```

FrmClassSchedule

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Configuration;
using System.Data.SqlClient;

namespace frmSplash
{
    public partial class frmClassSchedule : Form
    {
        SqlConnection connection;
        string connectionString;
        SqlDataReader DataReader;
        int A;
        int B;

        // Scheduled_Lessons Details
        List<int> ScheduleID = new List<int>();
        List<int> StudentID_Schedule = new List<int>();
        List<DateTime> ScheduleDate = new List<DateTime>();
        List<int> TeacherID_Schedule = new List<int>();
        List<string> ScheduledTime = new List<string>();

        // Student Details
        List<int> StudentID_Student = new List<int>();
        List<string> StudentFirstName = new List<string>();
        List<string> StudentLastName = new List<string>();
        List<int> InstrumentID_Students = new List<int>();

        // Teacher Details
        List<int> TeacherID_Teachers = new List<int>();
        List<string> TeacherFirstName = new List<string>();
        List<string> TeacherLastName = new List<string>();
        List<int> RoomID_Teachers = new List<int>();

        // Instrument Details
        List<int> InstrumentID_instruments = new List<int>();
        List<string> InstrumentName = new List<string>();

        // Desired Values
        List<int> DesiredTeacherIDs = new List<int>();
        List<string> DesiredStudentFirstName = new List<string>();
        List<string> DesiredStudentLastName = new List<string>();
        List<int> DesiredInstrumentID = new List<int>();
        List<int> DesiredScheduleIDs = new List<int>();
        List<int> DesiredStudentIDs = new List<int>();
        List<string> DesiredTeacherFirstName = new List<string>();
        List<string> DesiredTeacherLastName = new List<string>();
        List<string> DesiredInstrumentName = new List<string>();
        List<int> DesiredRoomID = new List<int>();
        List<string> DesiredScheduledTime = new List<string>();

        List<bool> DesiredDisplay = new List<bool>();
```

```
public frmClassSchedule()
{
    InitializeComponent();
    connectionString =
ConfigurationManager.ConnectionStrings["frmSplash.Properties.Settings.PrivateTuitionCo
nnnectionString"].ConnectionString;
}

private void frmClassSchedule_Load(object sender, EventArgs e)
{
    for (int x = 0; x < 6; x++)
    {
        DesiredDisplay.Add(false);
    }

    // This block of code will check the selected date, and update the format
    accodingly for visual astetics.
    if (GlobalVariables.SelectedDay == 1 || GlobalVariables.SelectedDay == 21
    || GlobalVariables.SelectedDay == 31)
    {
        lblDatePlaceholder.Text = String.Format("{0}st {1} {2}",
GlobalVariables.SelectedDay, GlobalVariables.SelectedMonthString,
GlobalVariables.SelectedYear);
    }
    else if (GlobalVariables.SelectedDay == 2 || GlobalVariables.SelectedDay
== 22)
    {
        lblDatePlaceholder.Text = String.Format("{0}nd {1} {2}",
GlobalVariables.SelectedDay, GlobalVariables.SelectedMonthString,
GlobalVariables.SelectedYear);
    }
    else if (GlobalVariables.SelectedDay == 3 || GlobalVariables.SelectedDay
== 23)
    {
        lblDatePlaceholder.Text = String.Format("{0}rd {1} {2}",
GlobalVariables.SelectedDay, GlobalVariables.SelectedMonthString,
GlobalVariables.SelectedYear);
    }
    else
    {
        lblDatePlaceholder.Text = String.Format("{0}th {1} {2}",
GlobalVariables.SelectedDay, GlobalVariables.SelectedMonthString,
GlobalVariables.SelectedYear);
    }
}

lblTimePlaceholder.Text = GlobalVariables.SelectedTimeTextFormat;

lblErrorMessage.Hide();
ScheduledLessons();
StudentDetails();
TeacherDetails();
InstrumentDetails();
AssignDatabaseValues();
DisplayInformaiton();
ErrorTest();
}

private void btnReturn_Click(object sender, EventArgs e)
{
```

```
GlobalVariables.SelectedTime = "";
GlobalVariables.SelectedDay = 0;
GlobalVariables.SelectedMonthInt = 0;
GlobalVariables.SelectedMonthString = "";
GlobalVariables.SelectedYear = 0;
GlobalVariables.SelectedScheduleDate = "00/00/0000";

A = 0;
B = 0;

frmPrivateTuition Main = new frmPrivateTuition();
Main.Show();
this.Hide();
}

public void ScheduledLessons()
{
    GlobalVariables.SelectedTime = GlobalVariables.SelectedTime.Substring(0,
5);
    using (connection = new SqlConnection(connectionString))
    {
        connection.Open();
        using (SqlCommand Command = new SqlCommand("SELECT * FROM
Scheduled_Lessons WHERE End_Date=@Date AND Booked_Time=@Time", connection))
        {
            Command.Parameters.AddWithValue("@Date",
Convert.ToDateTime(GlobalVariables.SelectedScheduleDate));
            Command.Parameters.AddWithValue("@Time",
GlobalVariables.SelectedTime);

            DataReader = Command.ExecuteReader();
            while (DataReader.Read())
            {
                DesiredScheduleIDs.Add(DataReader.GetInt32(0));
                DesiredStudentIDs.Add(DataReader.GetInt32(1));
                ScheduleDate.Add(DataReader.GetDateTime(5));
                DesiredTeacherIDs.Add(DataReader.GetInt32(2));
                DesiredScheduledTime.Add(DataReader.GetString(7));
            }
        }
    }
}

public void StudentDetails()
{
    using (connection = new SqlConnection(connectionString))
    {
        connection.Open();
        using (SqlCommand Command2 = new SqlCommand("SELECT * FROM Students",
connection))
        {

            DataReader = Command2.ExecuteReader();
            while (DataReader.Read())
            {
                StudentID_Student.Add(DataReader.GetInt32(0));
                StudentFirstName.Add(DataReader.GetString(1));
                StudentLastName.Add(DataReader.GetString(3));
                InstrumentID_Students.Add(DataReader.GetInt32(11));
            }
        }
    }
}
```

```
        }

    }

}

public void TeacherDetails()
{
    using (connection = new SqlConnection(connectionString))
    {
        connection.Open();
        using (SqlCommand Command3 = new SqlCommand("SELECT * FROM Teachers",
connection))
        {

            DataReader = Command3.ExecuteReader();
            while (DataReader.Read())
            {
                TeacherID_Teachers.Add(DataReader.GetInt32(0));
                TeacherFirstName.Add(DataReader.GetString(1));
                TeacherLastName.Add(DataReader.GetString(2));
                RoomID_Teachers.Add(DataReader.GetInt32(9));
            }
        }
    }
}

public void InstrumentDetails()
{
    using (connection = new SqlConnection(connectionString))
    {
        connection.Open();
        using (SqlCommand Command4 = new SqlCommand("SELECT * FROM
Instruments", connection))
        {

            DataReader = Command4.ExecuteReader();
            while (DataReader.Read())
            {
                InstrumentID_instruments.Add(DataReader.GetInt32(0));
                InstrumentName.Add(DataReader.GetString(2));
            }
        }
    }
}

private void btnCalender_Click(object sender, EventArgs e)
{
    GlobalVariables.SelectedTime = "";

    frmCalenderTimes LessonTimes = new frmCalenderTimes();
    LessonTimes.Show();
    this.Hide();
}

public void AssignDatabaseValues()
{
```

```
// Save Desired Student information
for (int x = 0; x < StudentID_Student.Count(); x++)
{
    for (int y = 0; y < DesiredStudentIDs.Count(); y++)
    {
        if (DesiredStudentIDs[y] == StudentID_Student[x])
        {
            DesiredStudentFirstName.Add(StudentFirstName[x]);
            DesiredStudentLastName.Add(StudentLastName[x]);
            DesiredInstrumentID.Add(InstrumentID_Students[x]);
        }
    }
}

// Save Desired Teacher Information
for (int x = 0; x < TeacherID_Teachers.Count(); x++)
{
    for (int y = 0; y < DesiredTeacherIDs.Count(); y++)
    {
        if (DesiredTeacherIDs[y] == TeacherID_Teachers[x])
        {
            DesiredTeacherFirstName.Add(TeacherFirstName[x]);
            DesiredTeacherLastName.Add(TeacherLastName[x]);
            DesiredRoomID.Add(RoomID_Teachers[x]);
        }
    }
}

// Save Desired Instrument Information
for (int x = 0; x < InstrumentID_instruments.Count(); x++)
{
    for (int y = 0; y < DesiredInstrumentID.Count(); y++)
    {
        if (DesiredInstrumentID[y] == InstrumentID_instruments[x])
        {
            DesiredInstrumentName.Add(InstrumentName[x]);
        }
    }
}

public void DisplayInformaiton()
{
    B = 0;

    for (int x = 0; x < DesiredScheduleIDs.Count(); x++)
    {
        if (DesiredDisplay[0] == false)
        {
            lblStudentName1.Text = DesiredStudentFirstName[x] + " " +
DesiredStudentLastName[x];
            lblTeacherName1.Text = DesiredTeacherFirstName[x] + " " +
DesiredTeacherLastName[x];
            lblInstrumentName1.Text = DesiredInstrumentName[x];
            lblRoomID1.Text = Convert.ToString(DesiredRoomID[x]);
            lblStudentName6.Text = x.ToString();
            DesiredDisplay[0] = true;
        }
        else if (DesiredDisplay[1] == false)
```

```
{  
    lblStudentName2.Text = DesiredStudentFirstName[x] + " " +  
DesiredStudentLastName[x];  
    lblTeacherName2.Text = DesiredTeacherFirstName[x] + " " +  
DesiredTeacherLastName[x];  
    lblInstrumentName2.Text = DesiredInstrumentName[x];  
    lblRoomID2.Text = Convert.ToString(DesiredRoomID[x]);  
    lblTeacherName6.Text = x.ToString();  
    DesiredDisplay[1] = true;  
}  
else if (DesiredDisplay[2] == false)  
{  
    lblStudentName3.Text = DesiredStudentFirstName[x] + " " +  
DesiredStudentLastName[x];  
    lblTeacherName3.Text = DesiredTeacherFirstName[x] + " " +  
DesiredTeacherLastName[x];  
    lblInstrumentName3.Text = DesiredInstrumentName[x];  
    lblRoomID3.Text = Convert.ToString(DesiredRoomID[x]);  
    lblInstrumentName6.Text = x.ToString();  
    DesiredDisplay[2] = true;  
}  
else if (DesiredDisplay[3] == false)  
{  
    lblStudentName4.Text = DesiredStudentFirstName[x] + " " +  
DesiredStudentLastName[x];  
    lblTeacherName4.Text = DesiredTeacherFirstName[x] + " " +  
DesiredTeacherLastName[x];  
    lblInstrumentName4.Text = DesiredInstrumentName[x];  
    lblRoomID4.Text = Convert.ToString(DesiredRoomID[x]);  
    DesiredDisplay[3] = true;  
}  
else if (DesiredDisplay[4] == false)  
{  
    lblStudentName5.Text = DesiredStudentFirstName[x] + " " +  
DesiredStudentLastName[x];  
    lblTeacherName5.Text = DesiredTeacherFirstName[x] + " " +  
DesiredTeacherLastName[x];  
    lblInstrumentName5.Text = DesiredInstrumentName[x];  
    lblRoomID5.Text = Convert.ToString(DesiredRoomID[x]);  
    DesiredDisplay[4] = true;  
}  
else if (DesiredDisplay[5] == false)  
{  
    lblStudentName6.Text = DesiredStudentFirstName[x] + " " +  
DesiredStudentLastName[x];  
    lblTeacherName6.Text = DesiredTeacherFirstName[x] + " " +  
DesiredTeacherLastName[x];  
    lblInstrumentName6.Text = DesiredInstrumentName[x];  
    lblRoomID6.Text = Convert.ToString(DesiredRoomID[x]);  
    DesiredDisplay[5] = true;  
}  
}  
  
if (DesiredDisplay[0] == false && DesiredDisplay.Count > 0)  
{  
    HideDisplay1();  
    HideDisplay2();  
    HideDisplay3();  
    HideDisplay4();  
    HideDisplay5();
```

```
        HideDisplay6();

        lblErrorMessage.Show();
        lblErrorMessage.Location = new Point(180, 268);
        lblErrorMessage.Text = "There are currently no scheduled" +
System.Environment.NewLine + "    classes for this date and time.";
    }

    if (DesiredDisplay[1] == false && DesiredDisplay.Count > 1)
    {
        HideDisplay2();
        HideDisplay3();
        HideDisplay4();
        HideDisplay5();
        HideDisplay6();
    }

    if (DesiredDisplay[2] == false && DesiredDisplay.Count > 2)
    {
        HideDisplay3();
        HideDisplay4();
        HideDisplay5();
        HideDisplay6();
    }

    if (DesiredDisplay[3] == false && DesiredDisplay.Count > 3)
    {
        HideDisplay4();
        HideDisplay5();
        HideDisplay6();
    }

    if (DesiredDisplay[4] == false && DesiredDisplay.Count > 4)
    {
        HideDisplay5();
        HideDisplay6();
    }

    if (DesiredDisplay[5] == false && DesiredDisplay.Count > 5)
    {
        HideDisplay6();
    }
}

public void HideDisplay1()
{
    pbDisplay1.Hide();
    lblStudentName1.Hide();
    lblTeacherName1.Hide();
    lblInstrumentName1.Hide();
    lblRoomID1.Hide();
}

public void HideDisplay2()
{
    pbDisplay2.Hide();
    lblStudentName2.Hide();
```

```
        lblTeacherName2.Hide();
        lblInstrumentName2.Hide();
        lblRoomID2.Hide();
    }

    public void HideDisplay3()
    {
        pbDisplay3.Hide();
        lblStudentName3.Hide();
        lblTeacherName3.Hide();
        lblInstrumentName3.Hide();
        lblRoomID3.Hide();
    }

    public void HideDisplay4()
    {
        pbDisplay4.Hide();
        lblStudentName4.Hide();
        lblTeacherName4.Hide();
        lblInstrumentName4.Hide();
        lblRoomID4.Hide();
    }

    public void HideDisplay5()
    {
        pbDisplay5.Hide();
        lblStudentName5.Hide();
        lblTeacherName5.Hide();
        lblInstrumentName5.Hide();
        lblRoomID5.Hide();
    }

    public void HideDisplay6()
    {
        pbDisplay6.Hide();
        lblStudentName6.Hide();
        lblTeacherName6.Hide();
        lblInstrumentName6.Hide();
        lblRoomID6.Hide();
    }

    public void ErrorTest()
    {
        A = 0;
        B = 0;

        while (A < DesiredScheduleIDs.Count)
        {
            while(B < DesiredTeacherIDs.Count)
            {
                int placeholder = DesiredTeacherIDs[A];

                if (DesiredTeacherIDs[A] == DesiredTeacherIDs[B] &&
DesiredTeacherIDs[B] != placeholder && GlobalVariables.scheduleErrorMessage == true)
                {
                    MessageBox.Show("A Teacher has been assigned to multiple lesson
for the same date and time. Please review schedule");
                }
            }
        }
    }
}
```

```
        }

        placeholder = DesiredRoomID[A];

        if(DesiredRoomID[A] == RoomID_Teachers[B] && DesiredRoomID[B] != placeholder && GlobalVariables.scheduleErrorMessage == true)
        {
            MessageBox.Show("A Room has been assigned multiple lessons for the same date and time. Please review schedule");
        }

        B++;
    }
    A++;
}
}

private void button1_Click(object sender, EventArgs e)
{
    // Load Add New Record Form
    GlobalVariables.PreviousForm = "Calender";
    frmAddField AddField = new frmAddField();
    AddField.Show();
    this.Hide();
}

public void LoadScheduleCalender()
{
    frmScheduleTable ScheduleTable = new frmScheduleTable();
    ScheduleTable.Show();
    this.Hide();
}

private void pictureBox1_Click(object sender, EventArgs e)
{
    GlobalVariables.CalendarReturn = true;
    GlobalVariables.CalenderStudentID = DesiredStudentIDs[0];
    GlobalVariables.CalenderScheduledID = DesiredScheduleIDs[0];
    LoadScheduleCalender();
}

private void pbDisplay2_Click(object sender, EventArgs e)
{
    GlobalVariables.CalendarReturn = true;
    GlobalVariables.CalenderStudentID = DesiredStudentIDs[1];
    GlobalVariables.CalenderScheduledID = DesiredScheduleIDs[1];
    LoadScheduleCalender();
}

private void pbDisplay3_Click(object sender, EventArgs e)
{
    GlobalVariables.CalendarReturn = true;
    GlobalVariables.CalenderStudentID = DesiredStudentIDs[2];
    GlobalVariables.CalenderScheduledID = DesiredScheduleIDs[2];
    LoadScheduleCalender();
}

private void pbDisplay4_Click(object sender, EventArgs e)
{
    GlobalVariables.CalendarReturn = true;
    GlobalVariables.CalenderStudentID = DesiredStudentIDs[3];
```

```
    GlobalVariables.CalenderScheduledID = DesiredScheduleIDs[3];
    LoadScheduleCalender();
}

private void pbDisplay5_Click(object sender, EventArgs e)
{
    GlobalVariables.CalendarReturn = true;
    GlobalVariables.CalenderStudentID = DesiredStudentIDs[4];
    GlobalVariables.CalenderScheduledID = DesiredScheduleIDs[4];
    LoadScheduleCalender();
}

private void pbDisplay6_Click(object sender, EventArgs e)
{
    GlobalVariables.CalendarReturn = true;
    GlobalVariables.CalenderStudentID = DesiredStudentIDs[5];
    GlobalVariables.CalenderScheduledID = DesiredScheduleIDs[5];
    LoadScheduleCalender();
}

}
```

FrmConfirmation

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;
using System.Configuration;

namespace frmSplash
{
    public partial class frmConfirmation : Form
    {
        SqlConnection connection;
        string connectionString;

        public frmConfirmation()
        {
            InitializeComponent();
            connectionString =
ConfigurationManager.ConnectionStrings["frmSplash.Properties.Settings.PrivateTuitionConnectionString"].ConnectionString;

            if (GlobalVariables.PreviousForm == "StudentTable")
            {
                DeleteStudentRecord();
            }
            else if (GlobalVariables.PreviousForm == "TeacherTable")
            {
                DeleteTeacherRecord();
            }
            else if (GlobalVariables.PreviousForm == "InstrumentTable")
            {
                DeleteInstrumentRecord();
            }
            else if (GlobalVariables.PreviousForm == "RoomTable")
            {
                DeleteRoomRecord();
            }
            else if (GlobalVariables.PreviousForm == "GradeTable")
            {
                DeleteGradeRecord();
            }
            else if (GlobalVariables.PreviousForm == "LessonBundleTable")
            {
                DeleteLessonBundleRecord();
            }
            else if (GlobalVariables.PreviousForm == "ScheduledLessonTable")
            {
                DeleteScheduledLessoRecord();
            }
            else if (GlobalVariables.PreviousForm == "PurchasedLessonsTable")
            {
                DeletePurchasedLessonRecord();
            }
        }
    }
}
```

```
public void DeleteStudentRecord()
{
    lblPlaceholder1.Text = "Student ID:";
    lblPlaceholder2.Text = "Student Name/";

    tbFieldID.Text = GlobalVariables.TableID.ToString();
    tbFieldName.Text = GlobalVariables.FieldNames;
}

public void DeleteTeacherRecord()
{
    lblPlaceholder1.Text = "Teacher ID:";
    lblPlaceholder2.Text = "Teacher Name/";

    tbFieldID.Text = GlobalVariables.TableID.ToString();
    tbFieldName.Text = GlobalVariables.FieldNames;
}

public void DeleteInstrumentRecord()
{
    lblPlaceholder1.Text = "Instrument ID:";
    lblPlaceholder2.Text = "Instrument Type-Name/";

    tbFieldID.Text = GlobalVariables.TableID.ToString();
    tbFieldName.Text = GlobalVariables.FieldNames;
}

public void DeleteRoomRecord()
{
    lblPlaceholder1.Text = "Room ID:";
    lblPlaceholder2.Text = "Room Specialisation/";

    tbFieldID.Text = GlobalVariables.TableID.ToString();
    tbFieldName.Text = GlobalVariables.FieldNames;
}

public void DeleteGradeRecord()
{
    lblPlaceholder1.Text = "Grade ID:";
    lblPlaceholder2.Text = "Grade Level/";

    tbFieldID.Text = GlobalVariables.TableID.ToString();
    tbFieldName.Text = GlobalVariables.FieldNames;
}

public void DeleteLessonBundleRecord()
{
    lblPlaceholder1.Text = "Lesson Bundle ID:";
    lblPlaceholder2.Text = "Lesson Bundle/";

    tbFieldID.Text = GlobalVariables.TableID.ToString();
    tbFieldName.Text = GlobalVariables.FieldNames;
}

public void DeleteScheduledLessoRecord()
{
    lblPlaceholder1.Text = "Scheduled Lesson ID:";
    lblPlaceholder2.Text = "Student Name";
```

```
        tbFieldID.Text = GlobalVariables.TableID.ToString();
        tbFieldName.Text = GlobalVariables.FieldNames;
    }

    public void DeletePurchasedLessonRecord()
    {
        lblPlaceholder1.Text = "Purchased Lesson ID:";  
        lblPlaceholder2.Text = "Student Name:";

        tbFieldID.Text = GlobalVariables.TableID.ToString();
        tbFieldName.Text = GlobalVariables.FieldNames;
    }

    public void ReturnToPreviousForm()
    {
        this.Hide();
    }

    private void btnCancel_Click(object sender, EventArgs e)
    {
        ReturnToPreviousForm();
    }

    private void btnConfirm_Click(object sender, EventArgs e)
    {
        if (GlobalVariables.PreviousForm == "StudentTable")
        {
            using (connection = new SqlConnection(connectionString))
            using (SqlCommand Studentcmd = new SqlCommand("DELETE FROM LessonsPurchased WHERE StudentID = @Value", connection))
            {
                connection.Open();
                Studentcmd.Parameters.AddWithValue("@Value",
Convert.ToInt32(tbFieldID.Text));
                Studentcmd.ExecuteNonQuery();
                connection.Close();
            }

            using (connection = new SqlConnection(connectionString))
            using (SqlCommand Studentcmd = new SqlCommand("DELETE FROM Scheduled_Lessons WHERE StudentID = @Value", connection))
            {
                connection.Open();
                Studentcmd.Parameters.AddWithValue("@Value",
Convert.ToInt32(tbFieldID.Text));
                Studentcmd.ExecuteNonQuery();
                connection.Close();
            }

            using (connection = new SqlConnection(connectionString))
            using (SqlCommand Studentcmd = new SqlCommand("DELETE FROM Students WHERE StudentID = @Value", connection))
            {
                connection.Open();
                Studentcmd.Parameters.AddWithValue("@Value",
Convert.ToInt32(tbFieldID.Text));
                Studentcmd.ExecuteNonQuery();
                connection.Close();
            }
        }
    }
}
```

```
        GlobalVariables.StudentForm.DisplayStudent();
        this.Hide();
    }
}
}
```

FrmUserLogin

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace frmSplash
{
    public partial class frmUserLogin : Form
    {

        public frmUserLogin()
        {
            InitializeComponent();
            lblError.Hide();
        }

        private void btnConfirm_Click(object sender, EventArgs e)
        {
            if (tbUsername.Text == "Admin" && tbPassword.Text == "Admin")
            {
                GlobalVariables.UserLoggedIn = true;
                GlobalVariables.Username = "Admin";
                CheckPreviousForm();
            }
            else
            {
                lblError.Show();
                lblError.Text = "Incorrect Details.";
            }
        }

        private void btnReturn_Click(object sender, EventArgs e)
        {
            CheckPreviousForm();
        }

        public void CheckPreviousForm()
        {
            if (GlobalVariables.PreviousForm == "StudentTable")
            {
                frmStudents StudentTable = new frmStudents();
                StudentTable.Show();
                this.Hide();
            }
            else if (GlobalVariables.PreviousForm == "TeacherTable")
            {
                frmTeachers TeacherTable = new frmTeachers();
                TeacherTable.Show();
                this.Hide();
            }
        }
    }
}
```

```
else if (GlobalVariables.PreviousForm == "InstrumentTable")
{
    frmInstrument InstrumentTable = new frmInstrument();
    InstrumentTable.Show();
    this.Hide();
}
else if (GlobalVariables.PreviousForm == "RoomTable")
{
    frmRoom RoomTable = new frmRoom();
    RoomTable.Show();
    this.Hide();
}
else if (GlobalVariables.PreviousForm == "GradeTable")
{
    frmGrade GradeTable = new frmGrade();
    GradeTable.Show();
    this.Hide();
}
else if (GlobalVariables.PreviousForm == "LessonBundleTable")
{
    frmLessonBundle LessonBundleTable = new frmLessonBundle();
    LessonBundleTable.Show();
    this.Hide();
}
else if (GlobalVariables.PreviousForm == "Menu")
{
    frmPrivateTuition Menu = new frmPrivateTuition();
    Menu.Show();
    this.Hide();
}
else if (GlobalVariables.PreviousForm == "ScheduledLessonTable")
{
    frmScheduleTable Schedule = new frmScheduleTable();
    Schedule.Show();
    this.Hide();
}
else if (GlobalVariables.PreviousForm == "PurchasedLessonBundleTable")
{
    PurchasedLessonBundles Purchase = new PurchasedLessonBundles();
    Purchase.Show();
    this.Hide();
}
}
```

FrmUpcomingLessons

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Configuration;
using System.Data.SqlClient;

namespace frmSplash
{
    public partial class frmViewUpcomingLessons : Form
    {
        SqlConnection connection;
        string connectionString;
        SqlDataReader DataReader;

        List<int> ScheduledLessons_ScheduledID = new List<int>();
        List<int> ScheduledLessons_StudentID = new List<int>();
        List<int> ScheduledLessons_TeacherID = new List<int>();
        List<int> ScheduledLessons_PurchasedID = new List<int>();
        List<string> ScheduledLessons_Time = new List<string>();

        List<int> Students_StudentID = new List<int>();
        List<string> Students_FirstName = new List<string>();
        List<string> Students_Surname = new List<string>();
        List<string> Students_FullName = new List<string>();

        List<int> Teacher_TeacherID = new List<int>();
        List<string> Teacher_FirstName = new List<string>();
        List<string> Teacher_Surname = new List<string>();
        List<string> Teacher_FullName = new List<string>();

        List<int> LessonDates_LessonDateID = new List<int>();
        List<int> LessonDates_ScheduleID = new List<int>();
        List<DateTime> LessonDates_Dates = new List<DateTime>();
        List<bool> LessonDates_Attended = new List<bool>();

        List<int> Display_LessonID = new List<int>();
        List<string> Display_Name = new List<string>();
        List<int> Display_PurchaseID = new List<int>();
        List<DateTime> Display_Date = new List<DateTime>();
        List<string> Display_Time = new List<string>();
        List<string> Display_Day = new List<string>();
        List<bool> Display_Attended = new List<bool>();

        List<int> Desired_ScheduleID = new List<int>();
        List<int> Desired_StudentID = new List<int>();
        List<int> Desired_TeacherID = new List<int>();

        List<int> PageTest = new List<int>();

        List<bool> Displays = new List<bool>();

        bool Student;
        string Day;
        DateTime Date;
```

```
int TotalPages;
int Additional_Records;

int Display_Start;
int Display_End;
string Search_Entity;

DateTime DateFilter;
DateTime DateFilter_Start;
DateTime DateFilter_End;
string DateFilter_String;
bool AssignPageNumber;
bool Limit;

int SelectedPage;
int MaxPage;
int StandardPageIncrment = 6;
intPageIndexReference;

bool[] HideDisplays = new bool[6];

public frmViewUpcomingLessons()
{
    InitializeComponent();
    connectionString =
ConfigurationManager.ConnectionStrings["frmSplash.Properties.Settings.PrivateTuitionCo
nnnectionString"].ConnectionString;
    label1.Text = "0 / 0";
    Display_Start = 0;
    Display_End = 5;
    Store_DisplayInformation();
    Search_Fill();
    Hide_Displays();
}

public void LocalStorage_Student()
{
    // This fuunction will be used to store various peices of information from
    the schedule Table. This will be used to complete the schedule
    using (connection = new SqlConnection(connectionString))
    {
        connection.Open();
        using (SqlCommand Command = new SqlCommand("SELECT * FROM Students",
connection))
        {
            DataReader = Command.ExecuteReader();
            while (DataReader.Read())
            {
                // For each record, store the corrisponding peice of
                information.
                Students_StudentID.Add(DataReader.GetInt32(0));
                Students_FirstName.Add(DataReader.GetString(1));
                Students_Surname.Add(DataReader.GetString(3));
            }
        }
        connection.Close();
    }

    for (int x = 0; x < Students_StudentID.Count(); x++)
    {
        Students_FullName.Add(Students_FirstName[x] + " " +
Students_Surname[x]);
    }
}
```

```
        }

    }

    public void LocalStorage_Teacher()
    {
        // This function will be used to store various pieces of information from
        // the schedule Table. This will be used to complete the schedule
        using (connection = new SqlConnection(connectionString))
        {
            connection.Open();
            using (SqlCommand Command = new SqlCommand("SELECT * FROM Teachers",
                connection))
            {
                DataReader = Command.ExecuteReader();
                while (DataReader.Read())
                {
                    // For each record, store the corresponding piece of
                    // information.
                    Teacher_TeacherID.Add(DataReader.GetInt32(0));
                    Teacher_FirstName.Add(DataReader.GetString(1));
                    Teacher_Surname.Add(DataReader.GetString(2));
                }
            }
            connection.Close();
        }

        for (int x = 0; x < Teacher_TeacherID.Count(); x++)
        {
            Teacher_FullName.Add(Teacher_FirstName[x] + " " + Teacher_Surname[x]);
        }
    }

    public void LocalStorage_LessonDates()
    {
        // This function will be used to store various pieces of information from
        // the schedule Table. This will be used to complete the schedule
        using (connection = new SqlConnection(connectionString))
        {
            connection.Open();
            using (SqlCommand Command = new SqlCommand("SELECT * FROM
                LessonDates", connection))
            {
                DataReader = Command.ExecuteReader();
                while (DataReader.Read())
                {
                    // For each record, store the corresponding piece of
                    // information.
                    LessonDates_LessonDateID.Add(DataReader.GetInt32(0));
                    LessonDates_ScheduleID.Add(DataReader.GetInt32(1));
                    LessonDates_Dates.Add(DataReader.GetDateTime(2));
                    LessonDates_Attended.Add(DataReader.GetBoolean(3));
                }
            }
        }

        for (int x = 0; x < LessonDates_LessonDateID.Count(); x++)
        {
            Display_LessonID.Add(LessonDates_LessonDateID[x]);
        }
    }
}
```

```
public void LocalStorage_ScheduledLessons()
{
    // This function will be used to store various pieces of information from
    // the schedule Table. This will be used to complete the schedule
    using (connection = new SqlConnection(connectionString))
    {
        connection.Open();
        using (SqlCommand Command = new SqlCommand("SELECT * FROM
Scheduled_Lessons", connection))
        {
            DataReader = Command.ExecuteReader();
            while (DataReader.Read())
            {
                // For each record, store the corresponding piece of
                // information.
                ScheduledLessons_ScheduledID.Add(DataReader.GetInt32(0));
                ScheduledLessons_StudentID.Add(DataReader.GetInt32(1));
                ScheduledLessons_TeacherID.Add(DataReader.GetInt32(2));
                ScheduledLessons_PurchasedID.Add(DataReader.GetInt32(3));
                ScheduledLessons_Time.Add(DataReader.GetString(7));
            }
        }
    }
}

public void LocalStorage_Clear()
{
    ScheduledLessons_ScheduledID.Clear();
    ScheduledLessons_StudentID.Clear();
    ScheduledLessons_TeacherID.Clear();
    ScheduledLessons_PurchasedID.Clear();
    ScheduledLessons_Time.Clear();

    Students_StudentID.Clear();
    Students_FirstName.Clear();
    Students_Surname.Clear();
    Students_FullName.Clear();

    Teacher_TeacherID.Clear();
    Teacher_FirstName.Clear();
    Teacher_Surname.Clear();
    Teacher_FullName.Clear();

    LessonDates_LessonDateID.Clear();
    LessonDates_ScheduleID.Clear();
    LessonDates_Dates.Clear();
    LessonDates_Attended.Clear();

    Display_LessonID.Clear();
    Display_Name.Clear();
    Display_PurchaseID.Clear();
    Display_Date.Clear();
    Display_Time.Clear();
    Display_Day.Clear();
    Display_Attended.Clear();

    Desired_ScheduleID.Clear();
    Desired_StudentID.Clear();
    Desired_TeacherID.Clear();
}
```

```
public void Search_Fill()
{
    if (Student == true)
    {
        Students_FullName.Sort();
        cbEntityID.DataSource = Students_FullName;
    }
    else if (Student == false)
    {
        Teacher_FullName.Sort();
        cbEntityID.DataSource = Teacher_FullName;
    }
}

private void btnSelectedEntity_Click(object sender, EventArgs e)
{
    if (btnSelectedEntity.Text == "Student")
    {
        Student = false;
        btnSelectedEntity.Text = "Teacher";
    }
    else if (btnSelectedEntity.Text == "Teacher")
    {
        Student = true;
        btnSelectedEntity.Text = "Student";
    }

    Store_DisplayInformation();
    Search_Fill();
}

private void cbEntityID_SelectedIndexChanged(object sender, EventArgs e)
{
    while (Displays.Count < 7)
    {
        Displays.Add(false);
    }
}

private void btnSearch_Click(object sender, EventArgs e)
{
    Store_DisplayInformation();
    Search_Fill();

    Search_Entity = cbEntityID.Text;

    // Locate Desired Student ID
    for (int x = 0; x < Display_Name.Count(); x++)
    {
        if (Display_Name[x] != Search_Entity)
        {
            Display_LessonID.RemoveAt(x);
            Display_PurchaseID.RemoveAt(x);
            Display_Name.RemoveAt(x);
            Display_Date.RemoveAt(x);
            Display_Day.RemoveAt(x);
            Display_Attended.RemoveAt(x);
        }
    }
}
```

```
        Display_Time.RemoveAt(x);
        x -= 1;
    }

    DateFilter_String = cbFilter.Text;
    if (DateFilter_String == "Today")
    {
        DateFilter = DateTime.Today;
    }
    else if (DateFilter_String == "This Week")
    {
    }
    else if (DateFilter_String == "This Month")
    {
        DateFilter = DateTime.Today;
        DateFilter = DateFilter.AddMonths(1);
    }

    DateFilter_Start = DateTime.Today;

    for (int x = 0; x < Display_Date.Count(); x++)
    {
        if (DateFilter_String == "Today")
        {
            if (Display_Date[x] != DateFilter)
            {
                Display_LessonID.RemoveAt(x);
                Display_PurchaseID.RemoveAt(x);
                Display_Name.RemoveAt(x);
                Display_Date.RemoveAt(x);
                Display_Day.RemoveAt(x);
                Display_Attended.RemoveAt(x);
                Display_Time.RemoveAt(x);
                x -= 1;
            }
        }
        else if (DateFilter_String == "This Week")
        {
            DateFilter_End = DateFilter_Start.AddDays(7);

            if (Display_Date[x] > DateFilter_Start && Display_Date[x] <
DateFilter_End)
            {
                Display_LessonID.RemoveAt(x);
                Display_PurchaseID.RemoveAt(x);
                Display_Name.RemoveAt(x);
                Display_Date.RemoveAt(x);
                Display_Day.RemoveAt(x);
                Display_Attended.RemoveAt(x);
                Display_Time.RemoveAt(x);
                x -= 1;
            }
        }
        else if (DateFilter_String == "This Month")
        {
            DateFilter_End = DateFilter_Start.AddMonths(1);
```

```
        if (Display_Date[x] > DateFilter_Start && Display_Date[x] <
DateFilter_End)
    {
        Display_LessonID.RemoveAt(x);
        Display_PurchaseID.RemoveAt(x);
        Display_Name.RemoveAt(x);
        Display_Date.RemoveAt(x);
        Display_Day.RemoveAt(x);
        Display_Attended.RemoveAt(x);
        Display_Time.RemoveAt(x);
        x -= 1;
    }

}
else if (DateFilter_String == "All Time")
{
    break;
}
}

for (int x = 0; x < Display_LessonID.Count(); x++)
{
    Displays.Add(false);
}

for (int x = 0; x < Displays.Count(); x++)
{
    Displays[x] = false;
}

TotalPages = 0;
SelectedPage = 1;
lblErrorMessage.Hide();
Additional_Records = 0;
PageIndexReference = 0;
AssignPageNumber = false;
Display_DisplayInformation();
AssignPageNumber = true;
Hide_Displays();
}

public void Store_DisplayInformation()
{
    LocalStorage_Clear();

    LocalStorage_Student();
    LocalStorage_Teacher();
    LocalStorage_ScheduledLessons();
    LocalStorage_LessonDates();

    for (int x = 0; x < LessonDates_LessonDateID.Count(); x++)
    {
        for (int y = 0; y < ScheduledLessons_ScheduledID.Count(); y++)
        {
            if (LessonDates_ScheduleID[x] == ScheduledLessons_ScheduledID[y])
            {
                Desired_ScheduleID.Add(LessonDates_LessonDateID[x]);
                Desired_StudentID.Add(ScheduledLessons_StudentID[y]);
                Desired_TeacherID.Add(ScheduledLessons_TeacherID[y]);
                Display_Date.Add(LessonDates_Dates[x]);
                Display_Attended.Add(LessonDates_Attended[x]);
            }
        }
    }
}
```

```
        Display_PurchaseID.Add(ScheduledLessons_PurchasedID[y]);
        Display_Time.Add(ScheduledLessons_Time[y]);
    }
}

for (int x = 0; x < Display_Date.Count(); x++)
{
    Date = Display_Date[x];
    Day = Date.DayOfWeek.ToString();

    Display_Day.Add(Day);
}

if (Student == true)
{
    for (int x = 0; x < Desired_StudentID.Count(); x++)
    {
        for (int y = 0; y < Students_StudentID.Count(); y++)
        {
            if (Desired_StudentID[x] == Students_StudentID[y])
            {
                Display_Name.Add(Students_FullName[y]);
            }
        }
    }
}
else if (Student == false)
{
    for (int x = 0; x < Desired_TeacherID.Count(); x++)
    {
        for (int y = 0; y < Teacher_TeacherID.Count(); y++)
        {
            if (Desired_TeacherID[x] == Teacher_TeacherID[y])
            {
                Display_Name.Add(Teacher_FullName[y]);
            }
        }
    }
}

public void Display_DisplayInformation()
{
    // Calculate total pages needed
    if (PageTest.Count() > 0)
    {
        PageTest.Clear();
    }

    for (int x = 0; x < Display_LessonID.Count(); x++)
    {
        PageTest.Add(1);
    }

    if (AssignPageNumber == false)
    {
        while (PageTest.Count() >= 6)
        {
            for (int x = 0; x < 6; x++)
            {

```

```
        PageTest.RemoveAt(0);
    }
    TotalPages++;
}

if (PageTest.Count() > 0)
{
    Additional_Records = PageTest.Count();
}

PageIndexReference = (SelectedPage * StandardPageIncrment);
PageIndexReference = PageIndexReference - 1;
}

label1.Text = (SelectedPage + " / " + TotalPages);

if (TotalPages > 0 && Additional_Records == 0)
{
    if (Displays[0] == false)
    {
        // 1st Display
        lblLessonID1.Show();
        lblEntityName1.Show();
        lblPurchasedLessonID1.Show();
        lblStartDate1.Show();
        lblBookedDays1.Show();
        lblBookedTime1.Show();
        checkAttended1.Show();
        pbDisplay1.Show();

        lblLessonID1.Text = Display_LessonID[(PageIndexReference -
5)].ToString();
        lblEntityName1.Text = Display_Name[(PageIndexReference - 5)];
        lblPurchasedLessonID1.Text =
Display_PurchaseID[(PageIndexReference - 5)].ToString();
        lblStartDate1.Text = Display_Date[(PageIndexReference -
5)].ToString();
        lblBookedDays1.Text = Display_Day[(PageIndexReference -
5)].ToString();
        lblBookedTime1.Text = Display_Time[(PageIndexReference -
5)].ToString();
        if (Display_Attended[(PageIndexReference - 5)] == true)
        {
            checkAttended1.CheckState = CheckState.Checked;
        }
        else
        {
            checkAttended1.CheckState = CheckState.Unchecked;
        }
        Displays[0] = true;
    }

    // 2nd Display
    if (Displays[1] == false)
    {
        lblLessonID2.Show();
        lblEntityName2.Show();
        lblPurchasedLessonID2.Show();
        lblStartDate2.Show();
        lblBookedDays2.Show();
        lblBookedTime2.Show();
    }
}
```

```
checkAttended2.Show();
pbDisplay2.Show();

lblLessonID2.Text = Display_LessonID[(PageIndexReference -
4)].ToString();
lblEntityName2.Text = Display_Name[(PageIndexReference - 4)];
lblPurchasedLessonID2.Text =
Display_PurchaseID[(PageIndexReference - 4)].ToString();
lblStartDate2.Text = Display_Date[(PageIndexReference -
4)].ToString();
lblBookedDays2.Text = Display_Day[(PageIndexReference -
4)].ToString();
lblBookedTime2.Text = Display_Time[(PageIndexReference -
4)].ToString();
if (Display_Attended[(PageIndexReference - 4)] == true)
{
    checkAttended2.CheckState = CheckState.Checked;
}
else
{
    checkAttended2.CheckState = CheckState.Unchecked;
}
Displays[1] = true;
}

// 3rd Display
if (Displays[2] == false)
{
    lblLessonID3.Show();
    lblEntityName3.Show();
    lblPurchasedLessonID3.Show();
    lblStartDate3.Show();
    lblBookedDays3.Show();
    lblBookedTime3.Show();
    checkAttended3.Show();
    pbDisplay3.Show();

    lblLessonID3.Text = Display_LessonID[(PageIndexReference -
3)].ToString();
    lblEntityName3.Text = Display_Name[(PageIndexReference - 3)];
    lblPurchasedLessonID3.Text =
Display_PurchaseID[(PageIndexReference - 3)].ToString();
    lblStartDate3.Text = Display_Date[(PageIndexReference -
3)].ToString();
    lblBookedDays3.Text = Display_Day[(PageIndexReference -
3)].ToString();
    lblBookedTime3.Text = Display_Time[(PageIndexReference -
3)].ToString();
    if (Display_Attended[(PageIndexReference - 3)] == true)
    {
        checkAttended3.CheckState = CheckState.Checked;
    }
    else
    {
        checkAttended3.CheckState = CheckState.Unchecked;
    }
    Displays[2] = true;
}

// 4th Display
```

```
        if (Displays[3] == false)
    {
        lblLessonID4.Show();
        lblEntityName4.Show();
        lblPurchasedLessonID4.Show();
        lblStartDate4.Show();
        lblBookedDays4.Show();
        lblBookedTime4.Show();
        checkAttended4.Show();
        pbDisplay4.Show();

        lblLessonID4.Text = Display_LessonID[(PageIndexReference -
2)].ToString();
        lblEntityName4.Text = Display_Name[(PageIndexReference - 2)];
        lblPurchasedLessonID4.Text =
Display_PurchaseID[(PageIndexReference - 2)].ToString();
        lblStartDate4.Text = Display_Date[(PageIndexReference -
2)].ToString();
        lblBookedDays4.Text = Display_Day[(PageIndexReference -
2)].ToString();
        lblBookedTime4.Text = Display_Time[(PageIndexReference -
2)].ToString();
        if (Display_Attended[(PageIndexReference - 2)] == true)
        {
            checkAttended4.CheckState = CheckState.Checked;
        }
        else
        {
            checkAttended4.CheckState = CheckState.Unchecked;
        }
        Displays[3] = true;
    }

    // 5th Display
    if (Displays[4] == false)
    {
        lblLessonID5.Show();
        lblEntityName5.Show();
        lblPurchasedLessonID5.Show();
        lblStartDate5.Show();
        lblBookedDays5.Show();
        lblBookedTime5.Show();
        checkAttended5.Show();
        pbDisplay5.Show();

        lblLessonID5.Text = Display_LessonID[(PageIndexReference -
1)].ToString();
        lblEntityName5.Text = Display_Name[(PageIndexReference - 1)];
        lblPurchasedLessonID5.Text =
Display_PurchaseID[(PageIndexReference - 1)].ToString();
        lblStartDate5.Text = Display_Date[(PageIndexReference -
1)].ToString();
        lblBookedDays5.Text = Display_Day[(PageIndexReference -
1)].ToString();
        lblBookedTime5.Text = Display_Time[(PageIndexReference -
1)].ToString();
        if (Display_Attended[(PageIndexReference - 1)] == true)
        {
            checkAttended5.CheckState = CheckState.Checked;
        }
        else
```

```
        {
            checkAttended5.CheckState = CheckState.Unchecked;
        }
    Displays[4] = true;
}

// 6th Display
if (Displays[5] == false)
{
    lblLessonID6.Show();
    lblEntityName6.Show();
    lblPurchasedLessonID6.Show();
    lblStartDate6.Show();
    lblBookedDays6.Show();
    lblBookedTime6.Show();
    checkAttended6.Show();
    pbDisplay6.Show();

    lblLessonID6.Text =
Display_LessonID[PageIndexReference].ToString();
    lblEntityName6.Text = Display_Name[PageIndexReference];
    lblPurchasedLessonID6.Text =
Display_PurchaseID[PageIndexReference].ToString();
    lblStartDate6.Text = Display_Date[PageIndexReference].ToString();
    lblBookedDays6.Text = Display_Day[PageIndexReference].ToString();
    lblBookedTime6.Text = Display_Time[PageIndexReference].ToString();
    if (Display_Attended[PageIndexReference] == true)
    {
        checkAttended6.CheckState = CheckState.Checked;
    }
    else
    {
        checkAttended6.CheckState = CheckState.Unchecked;
    }
    Displays[5] = true;
}
AssignPageNumber = true;
}
else if (TotalPages > 0 && Additional_Records > 0)
{
    if (Displays[0] == false)
    {
        // 1st Display
        lblLessonID1.Show();
        lblEntityName1.Show();
        lblPurchasedLessonID1.Show();
        lblStartDate1.Show();
        lblBookedDays1.Show();
        lblBookedTime1.Show();
        checkAttended1.Show();
        pbDisplay1.Show();

        lblLessonID1.Text = Display_LessonID[(PageIndexReference -
5)].ToString();
        lblEntityName1.Text = Display_Name[(PageIndexReference - 5)];
        lblPurchasedLessonID1.Text =
Display_PurchaseID[(PageIndexReference - 5)].ToString();
        lblStartDate1.Text = Display_Date[(PageIndexReference -
5)].ToString();
        lblBookedDays1.Text = Display_Day[(PageIndexReference -
5)].ToString();
    }
}
```

```
    lblBookedTime1.Text = Display_Time[(PageIndexReference - 5)].ToString();
    if (Display_Attended[(PageIndexReference - 5)] == true)
    {
        checkAttended1.CheckState = CheckState.Checked;
    }
    else
    {
        checkAttended1.CheckState = CheckState.Unchecked;
    }
    Displays[0] = true;
}

// 2nd Display
if (Displays[1] == false)
{
    lblLessonID2.Show();
    lblEntityName2.Show();
    lblPurchasedLessonID2.Show();
    lblStartDate2.Show();
    lblBookedDays2.Show();
    lblBookedTime2.Show();
    checkAttended2.Show();
    pbDisplay2.Show();

    lblLessonID2.Text = Display_LessonID[(PageIndexReference - 4)].ToString();
    lblEntityName2.Text = Display_Name[(PageIndexReference - 4)];
    lblPurchasedLessonID2.Text =
    Display_PurchaseID[(PageIndexReference - 4)].ToString();
    lblStartDate2.Text = Display_Date[(PageIndexReference - 4)].ToString();
    lblBookedDays2.Text = Display_Day[(PageIndexReference - 4)].ToString();
    lblBookedTime2.Text = Display_Time[(PageIndexReference - 4)].ToString();
    if (Display_Attended[(PageIndexReference - 4)] == true)
    {
        checkAttended2.CheckState = CheckState.Checked;
    }
    else
    {
        checkAttended2.CheckState = CheckState.Unchecked;
    }
    Displays[1] = true;
}

// 3rd Display
if (Displays[2] == false)
{
    lblLessonID3.Show();
    lblEntityName3.Show();
    lblPurchasedLessonID3.Show();
    lblStartDate3.Show();
    lblBookedDays3.Show();
    lblBookedTime3.Show();
    checkAttended3.Show();
    pbDisplay3.Show();

    lblLessonID3.Text = Display_LessonID[(PageIndexReference - 3)].ToString();
}
```

```
        lblEntityName3.Text = Display_Name[(PageIndexReference - 3)];
        lblPurchasedLessonID3.Text =
Display_PurchaseID[(PageIndexReference - 3)].ToString();
        lblStartDate3.Text = Display_Date[(PageIndexReference -
3)].ToString();
        lblBookedDays3.Text = Display_Day[(PageIndexReference -
3)].ToString();
        lblBookedTime3.Text = Display_Time[(PageIndexReference -
3)].ToString();
        if (Display_Attended[(PageIndexReference - 3)] == true)
{
    checkAttended3.CheckState = CheckState.Checked;
}
else
{
    checkAttended3.CheckState = CheckState.Unchecked;
}
Displays[2] = true;
}

// 4th Display
if (Displays[3] == false)
{
    lblLessonID4.Show();
    lblEntityName4.Show();
    lblPurchasedLessonID4.Show();
    lblStartDate4.Show();
    lblBookedDays4.Show();
    lblBookedTime4.Show();
    checkAttended4.Show();
    pbDisplay4.Show();

    lblLessonID4.Text = Display_LessonID[(PageIndexReference -
2)].ToString();
    lblEntityName4.Text = Display_Name[(PageIndexReference - 2)];
    lblPurchasedLessonID4.Text =
Display_PurchaseID[(PageIndexReference - 2)].ToString();
    lblStartDate4.Text = Display_Date[(PageIndexReference -
2)].ToString();
    lblBookedDays4.Text = Display_Day[(PageIndexReference -
2)].ToString();
    lblBookedTime4.Text = Display_Time[(PageIndexReference -
2)].ToString();
    if (Display_Attended[(PageIndexReference - 2)] == true)
{
    checkAttended4.CheckState = CheckState.Checked;
}
else
{
    checkAttended4.CheckState = CheckState.Unchecked;
}
Displays[3] = true;
}

// 5th Display
if (Displays[4] == false)
{
    lblLessonID5.Show();
    lblEntityName5.Show();
    lblPurchasedLessonID5.Show();
```

```
        lblStartDate5.Show();
        lblBookedDays5.Show();
        lblBookedTime5.Show();
        checkAttended5.Show();
        pbDisplay5.Show();

        lblLessonID5.Text = Display_LessonID[(PageIndexReference -
1)].ToString();
        lblEntityName5.Text = Display_Name[(PageIndexReference - 1)];
        lblPurchasedLessonID5.Text =
Display_PurchaseID[(PageIndexReference - 1)].ToString();
        lblStartDate5.Text = Display_Date[(PageIndexReference -
1)].ToString();
        lblBookedDays5.Text = Display_Day[(PageIndexReference -
1)].ToString();
        lblBookedTime5.Text = Display_Time[(PageIndexReference -
1)].ToString();
        if (Display_Attended[(PageIndexReference - 1)] == true)
        {
            checkAttended5.CheckState = CheckState.Checked;
        }
        else
        {
            checkAttended5.CheckState = CheckState.Unchecked;
        }
        Displays[4] = true;
    }

    // 6th Display
    if (Displays[5] == false)
    {
        lblLessonID6.Show();
        lblEntityName6.Show();
        lblPurchasedLessonID6.Show();
        lblStartDate6.Show();
        lblBookedDays6.Show();
        lblBookedTime6.Show();
        checkAttended6.Show();
        pbDisplay6.Show();

        lblLessonID6.Text =
Display_LessonID[PageIndexReference].ToString();
        lblEntityName6.Text = Display_Name[PageIndexReference];
        lblPurchasedLessonID6.Text =
Display_PurchaseID[PageIndexReference].ToString();
        lblStartDate6.Text = Display_Date[PageIndexReference].ToString();
        lblBookedDays6.Text = Display_Day[PageIndexReference].ToString();
        lblBookedTime6.Text = Display_Time[PageIndexReference].ToString();
        if (Display_Attended[PageIndexReference] == true)
        {
            checkAttended6.CheckState = CheckState.Checked;
        }
        else
        {
            checkAttended6.CheckState = CheckState.Unchecked;
        }
        Displays[5] = true;
    }

    Display_DisplayInformation_Extra();
    AssignPageNumber = true;
```

```
        }

        else if (TotalPages == 0 && Additional_Records > 0)
        {
            Display_DisplayInformation_Extra();
            AssignPageNumber = true;
        }
    }

    public void Display_DisplayInformation_Extra()
    {
        if (AssignPageNumber == false)
        {

           PageIndexReference = PageIndexReference + Additional_Records;
            TotalPages += 1;

            if (TotalPages == 1)
            {
                PageIndexReference -= 6;
            }
        }
    }

    if (SelectedPage == TotalPages)
    {
        if (Additional_Records == 5)
        {
            if (Displays[0] == false)
            {
                // 1st Display
                lblLessonID1.Show();
                lblEntityName1.Show();
                lblPurchasedLessonID1.Show();
                lblStartDate1.Show();
                lblBookedDays1.Show();
                lblBookedTime1.Show();
                checkAttended1.Show();
                pbDisplay1.Show();

                lblLessonID1.Text = Display_LessonID[(PageIndexReference -
4)].ToString();
                lblEntityName1.Text = Display_Name[(PageIndexReference - 4)];
                lblPurchasedLessonID1.Text =
Display_PurchaseID[(PageIndexReference - 4)].ToString();
                lblStartDate1.Text = Display_Date[(PageIndexReference -
4)].ToString();
                lblBookedDays1.Text = Display_Day[(PageIndexReference -
4)].ToString();
                lblBookedTime1.Text = Display_Time[(PageIndexReference -
4)].ToString();
                if (Display_Attended[(PageIndexReference - 4)] == true)
                {
                    checkAttended1.CheckState = CheckState.Checked;
                }
                else
                {
                    checkAttended1.CheckState = CheckState.Unchecked;
                }
                Displays[0] = true;
            }
        }
    }
}
```

```
        }
    // 2nd Display
    if (Displays[1] == false)
    {
        lblLessonID2.Show();
        lblEntityName2.Show();
        lblPurchasedLessonID2.Show();
        lblStartDate2.Show();
        lblBookedDays2.Show();
        lblBookedTime2.Show();
        checkAttended2.Show();
        pbDisplay2.Show();

        lblLessonID2.Text = Display_LessonID[(PageIndexReference -
3)].ToString();
        lblEntityName2.Text = Display_Name[(PageIndexReference - 3)];
        lblPurchasedLessonID2.Text =
Display_PurchaseID[(PageIndexReference - 3)].ToString();
        lblStartDate2.Text = Display_Date[(PageIndexReference -
3)].ToString();
        lblBookedDays2.Text = Display_Day[(PageIndexReference -
3)].ToString();
        lblBookedTime2.Text = Display_Time[(PageIndexReference -
3)].ToString();
        if (Display_Attended[(PageIndexReference - 3)] == true)
        {
            checkAttended2.CheckState = CheckState.Checked;
        }
        else
        {
            checkAttended2.CheckState = CheckState.Unchecked;
        }
        Displays[1] = true;
    }
    // 3rd Display
    if (Displays[2] == false)
    {
        lblLessonID3.Show();
        lblEntityName3.Show();
        lblPurchasedLessonID3.Show();
        lblStartDate3.Show();
        lblBookedDays3.Show();
        lblBookedTime3.Show();
        checkAttended3.Show();
        pbDisplay3.Show();

        lblLessonID3.Text = Display_LessonID[(PageIndexReference -
2)].ToString();
        lblEntityName3.Text = Display_Name[(PageIndexReference - 2)];
        lblPurchasedLessonID3.Text =
Display_PurchaseID[(PageIndexReference - 2)].ToString();
        lblStartDate3.Text = Display_Date[(PageIndexReference -
2)].ToString();
        lblBookedDays3.Text = Display_Day[(PageIndexReference -
2)].ToString();
        lblBookedTime3.Text = Display_Time[(PageIndexReference -
2)].ToString();
        if (Display_Attended[(PageIndexReference - 2)] == true)
        {
            checkAttended3.CheckState = CheckState.Checked;
        }
        else
```

```
        {
            checkAttended3.CheckState = CheckState.Unchecked;
        }
        Displays[2] = true;
    }
    // 4th Display
    if (Displays[3] == false)
    {
        lblLessonID4.Show();
        lblEntityName4.Show();
        lblPurchasedLessonID4.Show();
        lblStartDate4.Show();
        lblBookedDays4.Show();
        lblBookedTime4.Show();
        checkAttended4.Show();
        pbDisplay4.Show();

        lblLessonID4.Text = Display_LessonID[(PageIndexReference -
1)].ToString();
        lblEntityName4.Text = Display_Name[(PageIndexReference - 1)];
        lblPurchasedLessonID4.Text =
Display_PurchaseID[(PageIndexReference - 1)].ToString();
        lblStartDate4.Text = Display_Date[(PageIndexReference -
1)].ToString();
        lblBookedDays4.Text = Display_Day[(PageIndexReference -
1)].ToString();
        lblBookedTime4.Text = Display_Time[(PageIndexReference -
1)].ToString();
        if (Display_Attended[(PageIndexReference - 1)] == true)
        {
            checkAttended4.CheckState = CheckState.Checked;
        }
        else
        {
            checkAttended4.CheckState = CheckState.Unchecked;
        }
        Displays[3] = true;
    }
    // 5th Display
    if (Displays[4] == false)
    {
        lblLessonID5.Show();
        lblEntityName5.Show();
        lblPurchasedLessonID5.Show();
        lblStartDate5.Show();
        lblBookedDays5.Show();
        lblBookedTime5.Show();
        checkAttended5.Show();
        pbDisplay5.Show();

        lblLessonID5.Text =
Display_LessonID[(PageIndexReference)].ToString();
        lblEntityName5.Text = Display_Name[(PageIndexReference)];
        lblPurchasedLessonID5.Text =
Display_PurchaseID[(PageIndexReference)].ToString();
        lblStartDate5.Text =
Display_Date[(PageIndexReference)].ToString();
        lblBookedDays5.Text =
Display_Day[(PageIndexReference)].ToString();
        lblBookedTime5.Text =
Display_Time[(PageIndexReference)].ToString();
        if (Display_Attended[(PageIndexReference)] == true)
```

```
        {
            checkAttended5.CheckState = CheckState.Checked;
        }
        else
        {
            checkAttended5.CheckState = CheckState.Unchecked;
        }
        Displays[4] = true;
    }

}

else if (Additional_Records == 4)
{
    if (Displays[0] == false)
    {
        // 1st Display
        lblLessonID1.Show();
        lblEntityName1.Show();
        lblPurchasedLessonID1.Show();
        lblStartDate1.Show();
        lblBookedDays1.Show();
        lblBookedTime1.Show();
        checkAttended1.Show();
        pbDisplay1.Show();

        lblLessonID1.Text = Display_LessonID[(PageIndexReference -
3)].ToString();
        lblEntityName1.Text = Display_Name[(PageIndexReference - 3)];
        lblPurchasedLessonID1.Text =
Display_PurchaseID[(PageIndexReference - 3)].ToString();
        lblStartDate1.Text = Display_Date[(PageIndexReference -
3)].ToString();
        lblBookedDays1.Text = Display_Day[(PageIndexReference -
3)].ToString();
        lblBookedTime1.Text = Display_Time[(PageIndexReference -
3)].ToString();
        if (Display_Attended[(PageIndexReference - 3)] == true)
        {
            checkAttended1.CheckState = CheckState.Checked;
        }
        else
        {
            checkAttended1.CheckState = CheckState.Unchecked;
        }
        Displays[0] = true;
    }
    // 2nd Display
    if (Displays[1] == false)
    {
        lblLessonID2.Show();
        lblEntityName2.Show();
        lblPurchasedLessonID2.Show();
        lblStartDate2.Show();
        lblBookedDays2.Show();
        lblBookedTime2.Show();
        checkAttended2.Show();
        pbDisplay2.Show();

        lblLessonID2.Text = Display_LessonID[(PageIndexReference -
2)].ToString();
    }
}
```

```
        lblEntityName2.Text = Display_Name[(PageIndexReference - 2)];
        lblPurchasedLessonID2.Text =
Display_PurchaseID[(PageIndexReference - 2)].ToString();
        lblStartDate2.Text = Display_Date[(PageIndexReference -
2)].ToString();
        lblBookedDays2.Text = Display_Day[(PageIndexReference -
2)].ToString();
        lblBookedTime2.Text = Display_Time[(PageIndexReference -
2)].ToString();
        if (Display_Attended[(PageIndexReference - 2)] == true)
{
    checkAttended2.CheckState = CheckState.Checked;
}
else
{
    checkAttended2.CheckState = CheckState.Unchecked;
}
Displays[1] = true;
}
// 3rd Display
if (Displays[2] == false)
{
    lblLessonID3.Show();
    lblEntityName3.Show();
    lblPurchasedLessonID3.Show();
    lblStartDate3.Show();
    lblBookedDays3.Show();
    lblBookedTime3.Show();
    checkAttended3.Show();
    pbDisplay3.Show();

    lblLessonID3.Text = Display_LessonID[(PageIndexReference -
1)].ToString();
    lblEntityName3.Text = Display_Name[(PageIndexReference - 1)];
    lblPurchasedLessonID3.Text =
Display_PurchaseID[(PageIndexReference - 1)].ToString();
    lblStartDate3.Text = Display_Date[(PageIndexReference -
1)].ToString();
    lblBookedDays3.Text = Display_Day[(PageIndexReference -
1)].ToString();
    lblBookedTime3.Text = Display_Time[(PageIndexReference -
1)].ToString();
    if (Display_Attended[(PageIndexReference - 1)] == true)
{
    checkAttended3.CheckState = CheckState.Checked;
}
else
{
    checkAttended3.CheckState = CheckState.Unchecked;
}
Displays[2] = true;
}
// 4th Display
if (Displays[3] == false)
{
    lblLessonID4.Show();
    lblEntityName4.Show();
    lblPurchasedLessonID4.Show();
    lblStartDate4.Show();
    lblBookedDays4.Show();
    lblBookedTime4.Show();
    checkAttended4.Show();
```

```
        pbDisplay4.Show();

        lblLessonID4.Text =
Display_LessonID[(PageIndexReference)].ToString();
        lblEntityName4.Text = Display_Name[(PageIndexReference)];
        lblPurchasedLessonID4.Text =
Display_PurchaseID[(PageIndexReference)].ToString();
        lblStartDate4.Text =
Display_Date[(PageIndexReference)].ToString();
        lblBookedDays4.Text =
Display_Day[(PageIndexReference)].ToString();
        lblBookedTime4.Text =
Display_Time[(PageIndexReference)].ToString();
        if (Display_Attended[(PageIndexReference)] == true)
{
    checkAttended4.CheckState = CheckState.Checked;
}
else
{
    checkAttended4.CheckState = CheckState.Unchecked;
}
Displays[3] = true;
}

else if (Additional_Records == 3)
{
    if (Displays[0] == false)
{
    // 1st Display
    lblLessonID1.Show();
    lblEntityName1.Show();
    lblPurchasedLessonID1.Show();
    lblStartDate1.Show();
    lblBookedDays1.Show();
    lblBookedTime1.Show();
    checkAttended1.Show();
    pbDisplay1.Show();

    lblLessonID1.Text = Display_LessonID[(PageIndexReference -
2)].ToString();
    lblEntityName1.Text = Display_Name[(PageIndexReference - 2)];
    lblPurchasedLessonID1.Text =
Display_PurchaseID[(PageIndexReference - 2)].ToString();
    lblStartDate1.Text = Display_Date[(PageIndexReference -
2)].ToString();
    lblBookedDays1.Text = Display_Day[(PageIndexReference -
2)].ToString();
    lblBookedTime1.Text = Display_Time[(PageIndexReference -
2)].ToString();
    if (Display_Attended[(PageIndexReference - 2)] == true)
{
        checkAttended1.CheckState = CheckState.Checked;
}
else
{
        checkAttended1.CheckState = CheckState.Unchecked;
}
Displays[0] = true;
}
// 2nd Display
if (Displays[1] == false)
```

```
{  
    lblLessonID2.Show();  
    lblEntityName2.Show();  
    lblPurchasedLessonID2.Show();  
    lblStartDate2.Show();  
    lblBookedDays2.Show();  
    lblBookedTime2.Show();  
    checkAttended2.Show();  
    pbDisplay2.Show();  
  
    lblLessonID2.Text = Display_LessonID[(PageIndexReference -  
1)].ToString();  
    lblEntityName2.Text = Display_Name[(PageIndexReference - 1)];  
    lblPurchasedLessonID2.Text =  
Display_PurchaseID[(PageIndexReference - 1)].ToString();  
    lblStartDate2.Text = Display_Date[(PageIndexReference -  
1)].ToString();  
    lblBookedDays2.Text = Display_Day[(PageIndexReference -  
1)].ToString();  
    lblBookedTime2.Text = Display_Time[(PageIndexReference -  
1)].ToString();  
    if (Display_Attended[(PageIndexReference - 1)] == true)  
    {  
        checkAttended2.CheckState = CheckState.Checked;  
    }  
    else  
    {  
        checkAttended2.CheckState = CheckState.Unchecked;  
    }  
    Displays[1] = true;  
}  
// 3rd Display  
if (Displays[2] == false)  
{  
    lblLessonID3.Show();  
    lblEntityName3.Show();  
    lblPurchasedLessonID3.Show();  
    lblStartDate3.Show();  
    lblBookedDays3.Show();  
    lblBookedTime3.Show();  
    checkAttended3.Show();  
    pbDisplay3.Show();  
  
    lblLessonID3.Text =  
Display_LessonID[(PageIndexReference)].ToString();  
    lblEntityName3.Text = Display_Name[(PageIndexReference)];  
    lblPurchasedLessonID3.Text =  
Display_PurchaseID[(PageIndexReference)].ToString();  
    lblStartDate3.Text =  
Display_Date[(PageIndexReference)].ToString();  
    lblBookedDays3.Text =  
Display_Day[(PageIndexReference)].ToString();  
    lblBookedTime3.Text =  
Display_Time[(PageIndexReference)].ToString();  
    if (Display_Attended[(PageIndexReference)] == true)  
    {  
        checkAttended3.CheckState = CheckState.Checked;  
    }  
    else  
    {  
        checkAttended3.CheckState = CheckState.Unchecked;  
    }  
}
```

```
        Displays[2] = true;
    }

    else if (Additional_Records == 2)
    {
        if (Displays[0] == false)
        {
            // 1st Display
            lblLessonID1.Show();
            lblEntityName1.Show();
            lblPurchasedLessonID1.Show();
            lblStartDate1.Show();
            lblBookedDays1.Show();
            lblBookedTime1.Show();
            checkAttended1.Show();
            pbDisplay1.Show();

            lblLessonID1.Text = Display_LessonID[(PageIndexReference -
1)].ToString();
            lblEntityName1.Text = Display_Name[(PageIndexReference - 1)];
            lblPurchasedLessonID1.Text =
Display_PurchaseID[(PageIndexReference - 1)].ToString();
            lblStartDate1.Text = Display_Date[(PageIndexReference -
1)].ToString();
            lblBookedDays1.Text = Display_Day[(PageIndexReference -
1)].ToString();
            lblBookedTime1.Text = Display_Time[(PageIndexReference -
1)].ToString();
            if (Display_Attended[(PageIndexReference - 1)] == true)
            {
                checkAttended1.CheckState = CheckState.Checked;
            }
            else
            {
                checkAttended1.CheckState = CheckState.Unchecked;
            }
            Displays[0] = true;
        }
        // 2nd Display
        if (Displays[1] == false)
        {
            lblLessonID2.Show();
            lblEntityName2.Show();
            lblPurchasedLessonID2.Show();
            lblStartDate2.Show();
            lblBookedDays2.Show();
            lblBookedTime2.Show();
            checkAttended2.Show();
            pbDisplay2.Show();

            lblLessonID2.Text =
Display_LessonID[(PageIndexReference)].ToString();
            lblEntityName2.Text = Display_Name[(PageIndexReference)];
            lblPurchasedLessonID2.Text =
Display_PurchaseID[(PageIndexReference)].ToString();
            lblStartDate2.Text =
Display_Date[(PageIndexReference)].ToString();
            lblBookedDays2.Text =
Display_Day[(PageIndexReference)].ToString();
            lblBookedTime2.Text =
Display_Time[(PageIndexReference)].ToString();
        }
    }
}
```

```
        if (Display_Attended[(PageIndexReference)] == true)
    {
        checkAttended2.CheckState = CheckState.Checked;
    }
    else
    {
        checkAttended2.CheckState = CheckState.Unchecked;
    }
    Displays[1] = true;
}
}

else if (Additional_Records == 1)
{
    if (Displays[0] == false)
    {
        // 1st Display
        lblLessonID1.Show();
        lblEntityName1.Show();
        lblPurchasedLessonID1.Show();
        lblStartDate1.Show();
        lblBookedDays1.Show();
        lblBookedTime1.Show();
        checkAttended1.Show();
        pbDisplay1.Show();

        lblLessonID1.Text =
Display_LessonID[(PageIndexReference)].ToString();
        lblEntityName1.Text = Display_Name[(PageIndexReference)];
        lblPurchasedLessonID1.Text =
Display_PurchaseID[(PageIndexReference)].ToString();
        lblStartDate1.Text =
Display_Date[(PageIndexReference)].ToString();
        lblBookedDays1.Text =
Display_Day[(PageIndexReference)].ToString();
        lblBookedTime1.Text =
Display_Time[(PageIndexReference)].ToString();
        if (Display_Attended[(PageIndexReference)] == true)
        {
            checkAttended1.CheckState = CheckState.Checked;
        }
        else
        {
            checkAttended1.CheckState = CheckState.Unchecked;
        }
        Displays[0] = true;
    }
}

}
PageIndexReference = PageIndexReference - Additional_Records;
}
```

```
private void btnNext_Page_Click(object sender, EventArgs e)
{
    if (SelectedPage > 0 && SelectedPage != TotalPages)
    {
        SelectedPage += 1;
```

```
        for (int x = 0; x < Displays.Count(); x++)
    {
        Displays[x] = false;
    }
}

if (SelectedPage == TotalPages && Additional_Records > 0 &&
AssignPageNumber == true)
{
   PageIndexReference = (TotalPages - 1);
PageIndexReference = (PageIndexReference * StandardPageIncrment) +
Additional_Records;
PageIndexReference -= 1;

Display_DisplayInformation();
Hide_Displays();
}

private void btnPrevious_Page_Click(object sender, EventArgs e)
{
    if (SelectedPage == TotalPages && Additional_Records > 0 && SelectedPage
!= 1)
    {
       PageIndexReference = (TotalPages - 1);
PageIndexReference = (PageIndexReference * StandardPageIncrment);
PageIndexReference -= 1;
    }

    if (SelectedPage > 1)
    {
        SelectedPage -= 1;
        for (int x = 0; x < Displays.Count(); x++)
        {
            Displays[x] = false;
        }

        Display_DisplayInformation();
        Hide_Displays();
    }
}

private void btnReturn_Click(object sender, EventArgs e)
{
    frmPrivateTuition MainMenu = new frmPrivateTuition();
MainMenu.Show();
this.Close();
}

public void Hide_Displays()
{
    for (int x = 0; x < HideDisplays.Count(); x++)
    {
        HideDisplays[x] = false;
    }

    for (int x = 0; x < Display_Name.Count(); x++)
```

```
{  
    if (lblEntityName1.Text == Display_Name[x])  
    {  
        HideDisplays[0] = true;  
        break;  
    }  
  
    for (int x = 0; x < Display_Name.Count(); x++)  
    {  
        if (lblEntityName2.Text == Display_Name[x])  
        {  
            HideDisplays[1] = true;  
            break;  
        }  
  
        for (int x = 0; x < Display_Name.Count(); x++)  
        {  
            if (lblEntityName3.Text == Display_Name[x])  
            {  
                HideDisplays[2] = true;  
                break;  
            }  
  
            for (int x = 0; x < Display_Name.Count(); x++)  
            {  
                if (lblEntityName4.Text == Display_Name[x])  
                {  
                    HideDisplays[3] = true;  
                    break;  
                }  
  
                for (int x = 0; x < Display_Name.Count(); x++)  
                {  
                    if (lblEntityName5.Text == Display_Name[x])  
                    {  
                        HideDisplays[4] = true;  
                        break;  
                    }  
  
                    for (int x = 0; x < Display_Name.Count(); x++)  
                    {  
                        if (lblEntityName6.Text == Display_Name[x])  
                        {  
                            HideDisplays[5] = true;  
                            break;  
                        }  
  
                        if (HideDisplays[0] == false)  
                        {  
                            Hide_Display1();  
                            Hide_Display2();  
                            Hide_Display3();  
                            Hide_Display4();  
                            Hide_Display5();  
                            Hide_Display6();  
                        }  
                    }  
                }  
            }  
        }  
    }  
}
```

```
        lblErrorMessage.Show();
        lblErrorMessage.Location = new Point(264, 339);
        if (Student == true)
        {
            lblErrorMessage.Text = "No scheduled classes for this Student,
within this time period!";
        }
        else
        {
            lblErrorMessage.Text = "No scheduled classes for this Teacher,
within this time period!";
        }
    }
    if (HideDisplays[1] == false)
    {
        Hide_Display2();
        Hide_Display3();
        Hide_Display4();
        Hide_Display5();
        Hide_Display6();
    }
    if (HideDisplays[2] == false)
    {
        Hide_Display3();
        Hide_Display4();
        Hide_Display5();
        Hide_Display6();
    }
    if (HideDisplays[3] == false)
    {
        Hide_Display4();
        Hide_Display5();
        Hide_Display6();
    }
    if (HideDisplays[4] == false)
    {
        Hide_Display5();
        Hide_Display6();
    }
    if (HideDisplays[5] == false)
    {
        Hide_Display6();
    }
}

public void Hide_Display1()
{
    lblLessonID1.Hide();
    lblEntityName1.Hide();
    lblPurchasedLessonID1.Hide();
    lblStartDate1.Hide();
    lblBookedDays1.Hide();
    lblBookedTime1.Hide();
    checkAttended1.Hide();
    pbDisplay1.Hide();
}
public void Hide_Display2()
{
    lblLessonID2.Hide();
    lblEntityName2.Hide();
```

```
    lblPurchasedLessonID2.Hide();
    lblStartDate2.Hide();
    lblBookedDays2.Hide();
    lblBookedTime2.Hide();
    checkAttended2.Hide();
    pbDisplay2.Hide();
}
public void Hide_Display3()
{
    lblLessonID3.Hide();
    lblEntityName3.Hide();
    lblPurchasedLessonID3.Hide();
    lblStartDate3.Hide();
    lblBookedDays3.Hide();
    lblBookedTime3.Hide();
    checkAttended3.Hide();
    pbDisplay3.Hide();
}
public void Hide_Display4()
{
    lblLessonID4.Hide();
    lblEntityName4.Hide();
    lblPurchasedLessonID4.Hide();
    lblStartDate4.Hide();
    lblBookedDays4.Hide();
    lblBookedTime4.Hide();
    checkAttended4.Hide();
    pbDisplay4.Hide();
}
public void Hide_Display5()
{
    lblLessonID5.Hide();
    lblEntityName5.Hide();
    lblPurchasedLessonID5.Hide();
    lblStartDate5.Hide();
    lblBookedDays5.Hide();
    lblBookedTime5.Hide();
    checkAttended5.Hide();
    pbDisplay5.Hide();
}
public void Hide_Display6()
{
    lblLessonID6.Hide();
    lblEntityName6.Hide();
    lblPurchasedLessonID6.Hide();
    lblStartDate6.Hide();
    lblBookedDays6.Hide();
    lblBookedTime6.Hide();
    checkAttended6.Hide();
    pbDisplay6.Hide();
}
}
```

FrmlInvoiceStudentSelection

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Configuration;
using System.Data.SqlClient;

namespace frmSplash
{
    public partial class frmInvoiceStudentSelection : Form
    {
        SqlDataReader DataReader;

        SqlConnection connection;
        string connectionString;

        List<int> Students_StudentID = new List<int>();
        List<string> Students_FirstName = new List<string>();
        List<string> Students_Surname = new List<string>();
        List<string> Students_FullName_AND_StudentID = new List<string>();

        public frmInvoiceStudentSelection()
        {
            InitializeComponent();
            connectionString =
ConfigurationManager.ConnectionStrings["frmSplash.Properties.Settings.PrivateTuitionCo
nnectionString"].ConnectionString;
            Store_StudentInforamtion();
            SelectedStudentName();
        }

        public void Store_StudentInforamtion()
        {
            using (connection = new SqlConnection(connectionString))
            {
                connection.Open();
                using (SqlCommand Command = new SqlCommand("SELECT * FROM Students",
connection))
                {
                    DataReader = Command.ExecuteReader();
                    while (DataReader.Read())
                    {
                        // For each record, store the corrisponding peice of
information.
                        Students_StudentID.Add(DataReader.GetInt32(0));
                        Students_FirstName.Add(DataReader.GetString(1));
                        Students_Surname.Add(DataReader.GetString(3));
                    }
                }
                connection.Close();
            }

            for (int x = 0; x < Students_StudentID.Count(); x++)
            {
                Students_FullName_AND_StudentID.Add(Students_StudentID[x] + " |
" + Students_FirstName[x] + " " + Students_Surname[x]);
            }
        }
    }
}
```

```
        }
        comboBox1.DataSource = Students_FullName_AND_StudentID;
    }

    private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
    {
        string Sub;
        for (int x = 0; x < comboBox1.Text.Length; x++)
        {
            Sub = comboBox1.Text.Substring(x, 1);
            if (Sub == " ")
            {
                GlobalVariables.Invoice_SelectedStudentID =
Convert.ToInt32(comboBox1.Text.Substring(0, x));
                break;
            }
        }
    }

    public void SelectedStudentName()
    {
        for (int x = 0; x < Students_StudentID.Count(); x++)
        {
            if (GlobalVariables.Invoice_SelectedStudentID ==
Students_StudentID[x])
            {
                GlobalVariables.StudentName = Students_FirstName[x] + " " +
Students_Surname[x];
            }
        }
    }

    private void btnConfirm_Click(object sender, EventArgs e)
    {
        frmInvoiceReport Report = new frmInvoiceReport();
        Report.Show();
        this.Hide();
    }

    private void button1_Click(object sender, EventArgs e)
    {
        frmPrivateTuition Menu = new frmPrivateTuition();
        Menu.Show();
        this.Hide();
    }
}
```

FrmInvoiceReport

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Configuration;
using System.Data.SqlClient;

namespace frmSplash
{
    public partial class frmInvoiceReport : Form
    {
        SqlDataReader DataReader;

        SqlConnection connection;
        string connectionString;
        List<int> Purchase_IDs = new List<int>();
        List<int> Purchase_LessonBundle = new List<int>();
        List<DateTime> Purchase_PurchasedDate = new List<DateTime>();
        List<decimal> Purchase_BundleCost = new List<decimal>();
        List<bool> Displays = new List<bool>();
        double Total;

        Bitmap bmp;

        public frmInvoiceReport()
        {
            InitializeComponent();
            connectionString =
ConfigurationManager.ConnectionStrings["frmSplash.Properties.Settings.PrivateTuitionCo
nnnectionString"].ConnectionString;
        }

        private void frmInvoiceReport_Load(object sender, EventArgs e)
        {
            for (int x = 0; x < 6; x++)
            {
                Displays.Add(false);
            }

            lblErrorMessage.Hide();
            label3.Text = GlobalVariables.StudentName;

            Search();
            Fill_Display();
            Hide_ExcessInformation();
        }

        public void Search()
        {
            using (connection = new SqlConnection(connectionString))
            {
                connection.Open();
                using (SqlCommand Command = new SqlCommand("SELECT * FROM
LessonsPurchased WHERE StudentID = @StudentID AND Payment_Recieved = @Payment",
connection))
                {

```

```
        Command.Parameters.AddWithValue("@StudentID",
GlobalVariables.Invoice_SelectedStudentID);
        Command.Parameters.AddWithValue("@Payment", 0);

        DataReader = Command.ExecuteReader();
        while (DataReader.Read())
        {
            // For each record, store the corresponding piece of
information.
            Purchase_IDs.Add(DataReader.GetInt32(0));
            Purchase_LessonBundle.Add(DataReader.GetInt32(2));
            Purchase_PurchasedDate.Add(DataReader.GetDateTime(3));
            Purchase_BundleCost.Add(DataReader.GetDecimal(7));
        }
    }
connection.Close();
}

public void Fill_Display()
{
    for (int x = 0; x < Purchase_IDs.Count(); x++)
    {
        if (Displays[0] == false)
        {
            lblPurchasedLessonID1.Text = Purchase_IDs[x].ToString();
            lblStartDate1.Text = Purchase_LessonBundle[x].ToString();
            lblBookedDays1.Text =
Purchase_PurchasedDate[x].ToShortDateString();
            lblBookedTime1.Text = Purchase_BundleCost[x].ToString();
            Displays[0] = true;
        }
        else if (Displays[1] == false)
        {
            lblPurchasedLessonID2.Text = Purchase_IDs[x].ToString();
            lblStartDate2.Text = Purchase_LessonBundle[x].ToString();
            lblBookedDays2.Text =
Purchase_PurchasedDate[x].ToShortDateString();
            lblBookedTime2.Text = Purchase_BundleCost[x].ToString();
            Displays[1] = true;
        }
        else if (Displays[2] == false)
        {
            lblPurchasedLessonID3.Text = Purchase_IDs[x].ToString();
            lblStartDate3.Text = Purchase_LessonBundle[x].ToString();
            lblBookedDays3.Text =
Purchase_PurchasedDate[x].ToShortDateString();
            lblBookedTime3.Text = Purchase_BundleCost[x].ToString();
            Displays[2] = true;
        }
        else if (Displays[3] == false)
        {
            lblPurchasedLessonID4.Text = Purchase_IDs[x].ToString();
            lblStartDate4.Text = Purchase_LessonBundle[x].ToString();
            lblBookedDays4.Text =
Purchase_PurchasedDate[x].ToShortDateString();
            lblBookedTime4.Text = Purchase_BundleCost[x].ToString();
            Displays[3] = true;
        }
        else if (Displays[4] == false)
        {
            lblPurchasedLessonID5.Text = Purchase_IDs[x].ToString();
        }
    }
}
```

```
        lblStartDate5.Text = Purchase_LessonBundle[x].ToString();
        lblBookedDays5.Text =
Purchase_PurchasedDate[x].ToShortDateString();
        lblBookedTime5.Text = Purchase_BundleCost[x].ToString();
        Displays[4] = true;
    }
    else if (Displays[5] == false)
    {
        lblPurchasedLessonID6.Text = Purchase_IDs[x].ToString();
        lblStartDate6.Text = Purchase_LessonBundle[x].ToString();
        lblBookedDays6.Text =
Purchase_PurchasedDate[x].ToShortDateString();
        lblBookedTime6.Text = Purchase_BundleCost[x].ToString();
        Displays[5] = true;
    }

    Total = Total + Convert.ToInt32(Purchase_BundleCost[x]);
    lblTotal.Text = Total.ToString();
}

}

public void Hide_ExcessInformation()
{
    if (Displays[0] == false)
    {
        lblPurchasedLessonID1.Hide();
        lblStartDate1.Hide();
        lblBookedDays1.Hide();
        lblBookedTime1.Hide();
        pbDisplay1.Hide();
    }
    if (Displays[1] == false)
    {
        lblPurchasedLessonID2.Hide();
        lblStartDate2.Hide();
        lblBookedDays2.Hide();
        lblBookedTime2.Hide();
        pbDisplay2.Hide();
    }
    if (Displays[2] == false)
    {
        lblPurchasedLessonID3.Hide();
        lblStartDate3.Hide();
        lblBookedDays3.Hide();
        lblBookedTime3.Hide();
        pbDisplay3.Hide();
    }
    if (Displays[3] == false)
    {
        lblPurchasedLessonID4.Hide();
        lblStartDate4.Hide();
        lblBookedDays4.Hide();
        lblBookedTime4.Hide();
        pbDisplay4.Hide();
    }
    if (Displays[4] == false)
    {
        lblPurchasedLessonID5.Hide();
        lblStartDate5.Hide();
        lblBookedDays5.Hide();
        lblBookedTime5.Hide();
    }
}
```

```
        pbDisplay5.Hide();
    }
    if (Displays[5] == false)
    {
        lblPurchasedLessonID6.Hide();
        lblStartDate6.Hide();
        lblBookedDays6.Hide();
        lblBookedTime6.Hide();
        pbDisplay6.Hide();
    }

    if (Displays[0] == false && Displays[1] == false && Displays[2] == false
&& Displays[3] == false && Displays[4] == false && Displays[5] == false)
    {
        lblErrorMessage.Show();
        lblErrorMessage.Text = "This Student currently possess no outstanding
purchased lesson records.";
        lblErrorMessage.Location = new Point(46, 269);
    }
}

private void btnCalender_Click(object sender, EventArgs e)
{
    frmPrivateTuition Menu = new frmPrivateTuition();
    Menu.Show();
    this.Hide();
}

private void printDocument1_PrintPage(object sender,
System.Drawing.Printing.PrintEventArgs e)
{
    e.Graphics.DrawImage(bmp, 0, 0);
}

private void button1_Click(object sender, EventArgs e)
{
}

private void pictureBox2_Click(object sender, EventArgs e)
{
    Graphics g = this.CreateGraphics();
    bmp = new Bitmap(this.Size.Width, this.Size.Height, g);
    Graphics mg = Graphics.FromImage(bmp);
    mg.CopyFromScreen(this.Location.X, this.Location.Y, 0, 0, this.Size);
    printPreviewDialog1.ShowDialog();
}

// Print Button
private void pictureBox2_MouseEnter(object sender, EventArgs e)
{
    pictureBox2.BackColor = Color.LightGray;
    this.Cursor = Cursors.Hand;
}
private void pictureBox2_MouseLeave(object sender, EventArgs e)
{
    pictureBox2.BackColor = Color.PaleGreen;
    this.Cursor = Cursors.Arrow;
}
}
```

}

FrmGlobalVariables

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace frmSplash
{
    class GlobalVariables
    {
        // Calender Dates
        public static bool Date1;
        public static bool Date2;
        public static bool Date3;
        public static bool Date4;
        public static bool Date5;
        public static bool Date6;
        public static bool Date7;
        public static bool Date8;
        public static bool Date9;
        public static bool Date10;
        public static bool Date11;
        public static bool Date12;
        public static bool Date13;
        public static bool Date14;
        public static bool Date15;
        public static bool Date16;
        public static bool Date17;
        public static bool Date18;
        public static bool Date19;
        public static bool Date20;
        public static bool Date21;
        public static bool Date22;
        public static bool Date23;
        public static bool Date24;
        public static bool Date25;
        public static bool Date26;
        public static bool Date27;
        public static bool Date28;
        public static bool Date29;
        public static bool Date30;
        public static bool Date31;
        public static int SelectedDay;
        public static string DayName;

        // Calendar Months
        public static bool January;
        public static bool February;
        public static bool March;
        public static bool April;
        public static bool May;
        public static bool June;
        public static bool July;
        public static bool August;
        public static bool September;
        public static bool October;
```

```
public static bool November;
public static bool December;
public static int SelectedMonthInt;
public static string SelectedMonthString;

public static bool CalendarReturn;

// Time Variables
public static string SelectedTime;
public static string SelectedTimeTextFormat;

// Selected Year
public static int SelectedYear;

// Selected Schedule Date
public static string SelectedScheduleDate;

// Student Table
public static List<int> GlobalStudentID = new List<int>();

// Add Field Previous Form
public static string PreviousForm;
public static string Purpose;
public static string UpdateID;

// Student Record
public static string Student_FirstName;
public static string Student_OtherNames;
public static string Student_Surname;
public static string Student_DOB;
public static string Student_Address;
public static string Student_Town;
public static string Student_PostCode;
public static string Student_ContactNumber;
public static string Student_Email;
public static string Student_GradeID;
public static string Student_InstrumentID;
public static string Student_PaymentFee;

// Teacher Record
public static string Teacher_FirstName;
public static string Teacher_Surname;
public static string Teacher_Address;
public static string Teacher_Town;
public static string Teacher_PostCode;
public static string Teacher_ContactNumber;
public static string Teacher_Email;
public static string Teacher_Specialisation;
public static string Teacher_RoomID;

// Instrument Record
public static string Instrument_Type;
public static string Instrument_Name;
public static string Instrument_Quantity;

// Room Record
public static string Room_Type;
public static string Room_Specialisation;

// Grade Record
public static string Grade_Level;
public static string Grade_Fee;
```

```
// LessonBundle Record
public static string Bundle_Name;
public static string Bundle_Cost;
public static string Bundle_Discount;

// Scheudled Lesson Record
public static string Schedule_StudentID;
public static string Schedule_TeacherID;
public static string Schedule_PurchaseID;
public static string Schedule_Weeks;
public static string Schedule_StartDate;
public static string Schedule_BookedDays;
public static string Schedule_BookedTime;
public static string Schedule_EndDate;

// Purchased Lesson Record
public static string Purchase_StudentID;
public static string Purchase_BundleID;
public static string Purchase_Date;
public static string Purchase_Method;
public static string Purchase_Recieved;
public static string Purchase_ReceivedDate;
public static string Purchase_BundleCosts;

// Canender selected Varaibles
public static int CalenderStudentID;
public static int CalenderScheduledID;

public static bool scheduleErrorMessage;

// Admin Login
public static string Username;
public static bool UserLoggedIn;

// Delete Details
public static int TableID;
public static string FieldNames;

// Global Form Entities
public static frmStudents StudentForm = new frmStudents();
public static frmTeachers TeacherForm = new frmTeachers();
public static frmInstrument InstrumentForm = new frmInstrument();
public static frmLessonBundle LessonBundleForm = new frmLessonBundle();
public static frmGrade GradeForm = new frmGrade();
public static frmRoom RoomForm = new frmRoom();
public static frmScheduleTable ScheduledLessonForm = new frmScheduleTable();
public static PurchasedLessonBundles PurchasedLessonForm = new
PurchasedLessonBundles();
public static frmCalenderDates Calendar = new frmCalenderDates();
public static frmAddField AddField = new frmAddField();

// Invoice
public static int Invoice_SelectedStudentID;
public static string StudentName;
}
```