

INTERVAL TRAINING SOFTWARE NEA

Catherine Taylor, Student Number: 2344, Centre Number: 64395

GODALMING COLLEGE

Contents

Research.....	3
Background	3
Interview	4
List of Questions.....	4
Transcript	5
Summary	6
Observation.....	6
Questionnaire	7
The Google Form.....	7
Responses	8
Summary	13
Document Capture.....	13
Research Summary	16
Analysis	17
Description of Problem	17
Description of Current System.....	17
Analogue System Description and Flow Chart.....	17
Digital System Description and Flow Chart.....	19
Systems Outline Chart for Proposed System	21
Prospective Users.....	22
User Needs and Acceptable Limitations	22
Data Sources and Destinations	23
Volumetrics and Data Dictionary	24
Data Flow Diagrams	27
Existing System.....	27
Proposed System	28
Requirements.....	29
Requirements Dialogue.....	30
Analysis Summary	32
Design.....	33
Object Analysis Diagrams.....	33
User Interface	34
Navigation Diagrams	39

Main Menus	40
Student Menu	41
Teacher Menu	42
Data Structures/Storage	43
Database Normalisation.....	43
E-R Diagram.....	44
Storage Specifications	45
SQL DDL.....	48
Processes.....	50
Level Based Test Generation.....	50
Generating Starting Note and Direction	51
Previous Results Based Test Generation.....	52
Validation	54
SQL DML.....	55
Communication.....	58
Database Connection	58
Testing Strategy	59
Processing Test Data	59
Process Testing Plan.....	60
GUI/IO Test Data	61
GUI/IO Testing Plan.....	62
Beta Testing	63
Testing.....	64
GUI/IO Testing.....	64
Process Testing	66
Beta Testing	69
Evaluation	70
Revisiting the Problem	70
Additional Features.....	70
Achievement of Requirements	71
Conclusion.....	74
Bibliography	75
Appendix	77
User Interface	77
Code	80

Research

Background

The aim of this project is to create a program for the Godalming College music department that tests the students' knowledge and identification of musical intervals. For music A level, students need a clear understanding and recognition of intervals in order to excel in listening papers, composition, and performances. The system will be used by both teachers and students.

An interval is the name for the difference in pitch between two notes. Each interval has its own unique name within an octave. The name for each interval is comprised of two parts: its quality (major, minor, perfect, augmented, and diminished) and its number (unison, 2nd, 3rd, 4th, 5th, 6th, 7th, octave). This naming method continues until an octave – any interval larger than an octave is then identified in the same way but with “compound” in front of it or it can continue on as 9th, 10th, 11th etc.

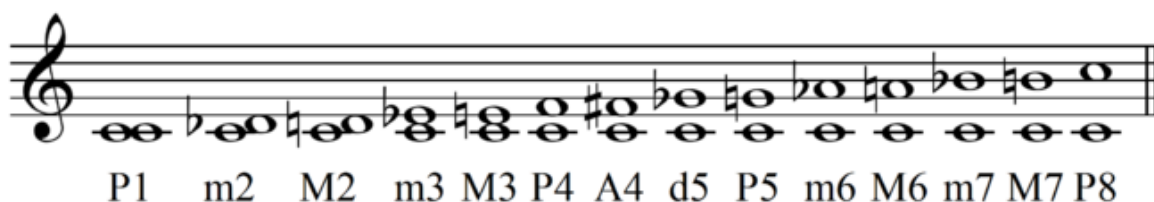


Figure 1 - Score notation of the intervals from C

Number of semitones	Minor, major, or perfect intervals	Short	Augmented or diminished intervals	Short
0	Perfect unison ^{[5][b]}	P1	Diminished second	d2
1	Minor second	m2	Augmented unison ^{[5][b]}	A1
2	Major second	M2	Diminished third	d3
3	Minor third	m3	Augmented second	A2
4	Major third	M3	Diminished fourth	d4
5	Perfect fourth	P4	Augmented third	A3
6			Diminished fifth	d5
			Augmented fourth	A4
7	Perfect fifth	P5	Diminished sixth	d6
8	Minor sixth	m6	Augmented fifth	A5
9	Major sixth	M6	Diminished seventh	d7
10	Minor seventh	m7	Augmented sixth	A6
11	Major seventh	M7	Diminished octave	d8
12	Perfect octave	P8	Augmented seventh	A7

Figure 2 - Table of the number of semitones in an interval and its corresponding name

The current system in place is a mixture of digital and analogue learning. During the lessons, the teacher will play an interval on the piano and then the students in the class will write down the name of the interval and then the teacher will give the correct answers. This system is effective as the students can receive teacher feedback and the teacher can decide what intervals the students are being tested on. However, the issue with this system is that it does not store the students' answers and results and the intervals that students have struggled with are not noted down. The online system is a piece of software called Auralia. This enables students to test themselves on intervals however it lacks the teacher feedback, and the teachers cannot track the progress of the students.

The problem with the current systems is that there is not the ability for teachers to track the learning of the students and ensure that students have mastered each interval. Also, currently with the analogue system the teachers do not store the results of each quiz.

Currently the problem affects both teachers and students. It affects teachers as they cannot easily track and view the progress of students and whether or not students are using the practise software making it harder to know which students may need help or extra teaching. It affects the students as there is no way for them to identify the areas they struggle with in the analogue system and there is little incentive to use the software as there is no teacher feedback which can make doing the practise seem pointless.

Interview

I interviewed in person **Paul Clifford** on Tuesday 15th September at 2:30pm. Paul teaches A level music at Godalming College so is third party for my project and the primary user. I asked him about the current system and also asked him about what he would want for the new system.

List of Questions

- 1) What is and what is wrong with the current analogue system?
- 2) What is and what is wrong with the current digital system?
- 3) Which range intervals would you like to be tested?
- 4) What instrument should play the intervals, or should there be a variety?
- 5) What platform would you like the software to be for?
- 6) How would you like the system to function? Would you like just one interval tested multiple times until learnt or a variety in each test?
- 7) Would you like the ability to customise a test in order to test students on the intervals you want?
- 8) Would you like an option for a randomly generated test?
- 9) Would you like a summary page for each student which details their attempts at identifying intervals? How would you like it displayed? E.g., graph, percentage, list of answers
- 10) How would you like students to input the answers? E.g., multiple choice, multiple choice for each quality and number, text box or a combination
- 11) Would you like different difficulty settings? If so, how should they be decided?

Transcript

- 1) What is and what is wrong with the current analogue system?

In a lesson I will play intervals on the piano and ask the students to recognise them. Generally, I do not write down students' answers to the questions or take down their mark so it can be difficult to track the progress of students.

- 2) What is and what is wrong with the current digital system?

We have an aural software called Auralia on the music department computers where the students can test themselves on different difficulty levels and it tracks their progress. The problem with Auralia is it is purely student based and there is no teacher interaction element so the students rarely use it, and I cannot see their progress even if it is being tracked.

- 3) Which range intervals would you like to be tested?

I would like all intervals from a 2nd to a 15th, including major, minor, perfect, augmented, and diminished. I would also like for a mixture of both ascending and descending intervals to be tested.

- 4) What instrument should play the intervals, or should there be a variety?

Just a computer-generated sound should be fine for the sound. It will be easier for students to concentrate on the pitch rather than associating the sound with an instrument.

- 5) What platform would you like the software to be for?

I would like it to be able to be downloaded on the Windows PCs in the music department.

- 6) How would you like the system to function? Would you like just one interval tested multiple times until learnt or a variety in each test?

I would like each test to have 10 different intervals with the ability to view the ones that they have got incorrect at the end of the test.

- 7) Would you like the ability to customise a test in order to test students on the intervals you want?

Yes, I would. It will be useful for me to set them specific intervals that I think might be more challenging or that students might need practise on.

- 8) Would you like an option for a randomly generated test?

Yes, it would be great for students to practise in their own time and get random new intervals.

- 9) Would you like a summary page for each student which details their attempts at identifying intervals? How would you like it displayed? E.g., graph, percentage, list of answers

I would like the students' average overall test score to be displayed to me as a percentage. I would like a graph that displays each student's average percentage in tests so that I can easily see if they are improving over time.

10) How would you like students to input the answers? E.g., multiple choice, multiple choice for each quality and number, text box or a combination

I would like the students to input the answer from a multiple choice for the quality and number separately where every quality and number is listed.

11) Would you like different difficulty settings? If so, how should they be decided?

Yes, I would like different difficulty settings. I would like the easiest setting to be where it can play the easiest intervals to identify which are: 2nds, 3rds, and perfect 4ths, 5ths, and 8ths. The next difficulty up should be all major, minor, and perfect intervals and you have to identify the number and the quality. The next level should be identifying major, minor, perfect, augmented, and diminished intervals. The hardest level should include compound intervals as well as identifying major, minor, perfect, augmented, and diminished.

Summary

Overall, Paul wants a system that is built for Windows PC that tests the intervals from 2nd to 15th including major, minor, perfect, augmented, diminished and compound intervals. Each test should have 10 questions and have a range of difficulty levels determined by the difficulty of the interval. He wants the tests to be able to be customised and to be able to have randomly generated tests as well. He would also like a graph and percentage to be generated for each student which he can view.

Observation

I have been in Paul's music A level lessons starting from September 2019 and have observed when we were tested on intervals in the music classroom at Godalming College.

Paul will play an interval on the piano, making sure that the students cannot see the piano keys. Once he has played the interval twice, he gives the students between 30 seconds and 1 minute to work out the interval and write down its name. If requested, he will play the interval again. After the time has run out, he goes around the room and asks each student what they got as their answer and then he says the correct answer. He then asks the students to mark their answer and then moves on to the next interval. He would usually ask 10 intervals each test. At the end of the lesson the students will file away their answers that they have written down on paper in their folders.


I have also used the software Auralia in music lesson time since September 2019 in the music computer room at Godalming College.

Auralia requires a unique username and password for each student to log in. The student then clicks on the interval recognition option. It then has a selection of difficulty levels for you to choose from ranging from level 1 to level 17. The software then plays the interval using a piano sound and the user can then select the name of the interval from a multiple choice, or input it using a keyboard or guitar. It will then instantly mark your answer and tell you if you have got it correct or not – you can then replay the interval to consolidate. The program then tracks your percentage score for this practise session as well as your average percentage score overall.

Questionnaire

I created a questionnaire in Google Forms on the 15th of September which I sent out to all of the music A level students as they will be the secondary users for the software. I asked them about how frequently they use the current system, what they like and dislike about the current system and what they would like in the new system. I also asked them what year group they are in in order to take account for the fact that the L6 have not been at college long and may not know the system as well.

The Google Form



MUSIC STUDENT SURVEY

I am developing a interval testing and recognition software for the music department for my computer science coursework and I would really appreciate you answering these questions. Thanks, Katie Taylor :)

**Required*

What year are you in? *

☐ L6

☐ U6

Do you feel as though you need to practise your interval recognition?

☐ Yes

☐ No

Are you aware that the music department has a digital interval recognition software, Auralia, that you can use to practise interval recognition? *

☐ Yes

☐ No

If you were aware of it - have you used it and how frequently do you use it?

☐ Yes - I use it regularly

☐ Yes - I use it occasionally

☐ Yes - I have used it once or twice

☐ No - I have not used it

If you have used it - what do you like/dislike about it?

Your answer

What do you like/dislike about being tested on intervals in class and writing answers on paper? *

Your answer

Do you feel like the current system (either in class questions or Auralia) is tracking your progress? *

☐ Yes

☐ No

Tick the top three things you would want in an interval testing software *

☐ The ability for teachers to create tests, assign them to you, and see your score

☐ The ability for you to create your own randomly generated tests to test your interval recognition ability

☐ A graph that shows your progress and if your interval recognition has improved

☐ The ability to choose between easy, medium and hard testing modes

☐ A range of instrument sounds playing the intervals for you to recognise

☐ Multiple testing modes such as multiple choice and text boxes

☐ The ability to see the intervals you have misidentified the most and test yourself on those

Are there any other features that you would like to suggest? *

Your answer

Submit

Figure 3 - Screenshots of Google Form sent to music students.

Responses

Multiple Choice

What year are you in?

19 responses

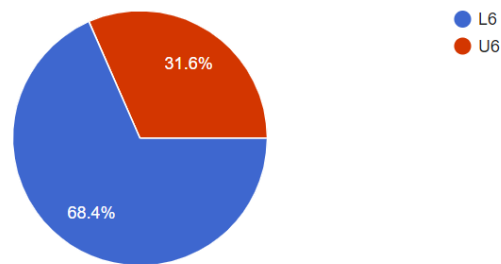


Figure 4 - Questionnaire Q1

I asked this question in order to know how skewed the results may be due to the fact that the L6 have only been at college for a few months and therefore will be less familiar with the systems in place. The results showed that two thirds of the students were L6 that took the questionnaire, so I have taken this into account when looking at results.

Do you feel as though you need to practise your interval recognition?

19 responses

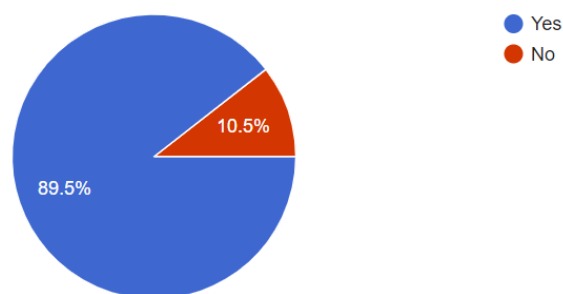


Figure 5 - Questionnaire Q2

I asked this question to assess the necessity of the new program and how important interval recognition is perceived to be by the A Level students. From the results it is clear that the overwhelming majority of the students believe that they need interval recognition practice which confirms that my project will be useful for them.

Do you feel like the current system (either in class questions or Auralia) is tracking your progress?

19 responses

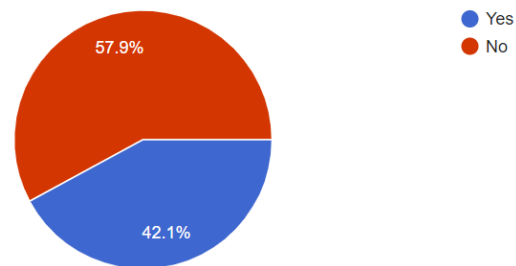


Figure 6 - Questionnaire Q3

This question was also asked to gauge how needed the program was. Considering that the majority felt that the current system in place is not effective in tracking progress it therefore indicates that my project will be useful and will help the students.

Are you aware that the music department has a digital interval recognition software, Auralia, that you can use to practise interval recognition?

19 responses

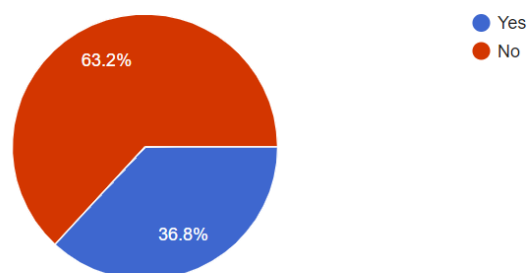


Figure 7 - Questionnaire Q4

The majority of students who answered the questionnaire had not heard of the software currently in place, Auralia. This might be due to the fact that Paul has not told them about it due to the fact that in the past when he has told students about it they have never used it and therefore there would be no need to tell them about it. This means that the current system does need to be changed.

If you were aware of it - have you used it and how frequently do you use it?

18 responses

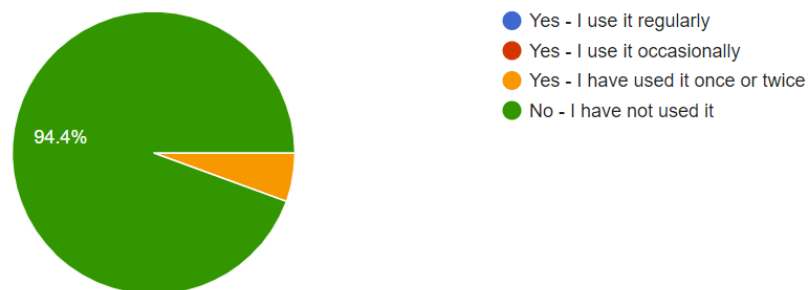


Figure 8 - Questionnaire Q5

This question shows how much students have used the software if they were aware of it. It shows that no students use the software occasionally or regularly, a very small number have used it once or twice and the majority have not used it. This shows that students have no desire to use the software and even if they do use it, it is not good enough to make them use it more than once or twice which highlights the need for a new system that provides an incentive for students to use it.

Tick the top three things you would want in an interval testing software

19 responses

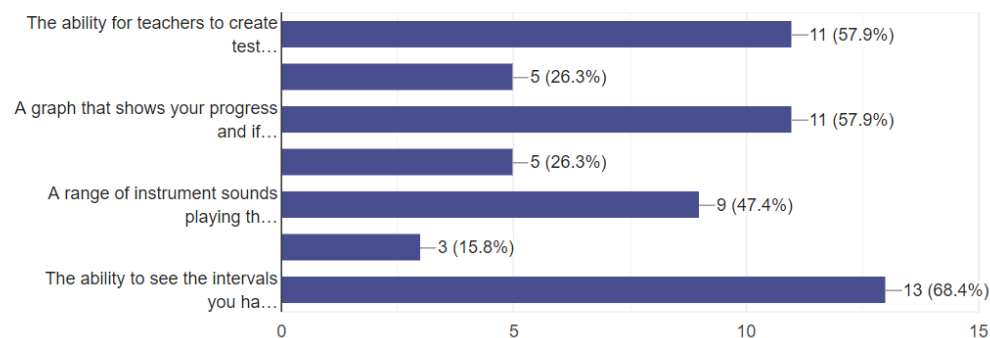


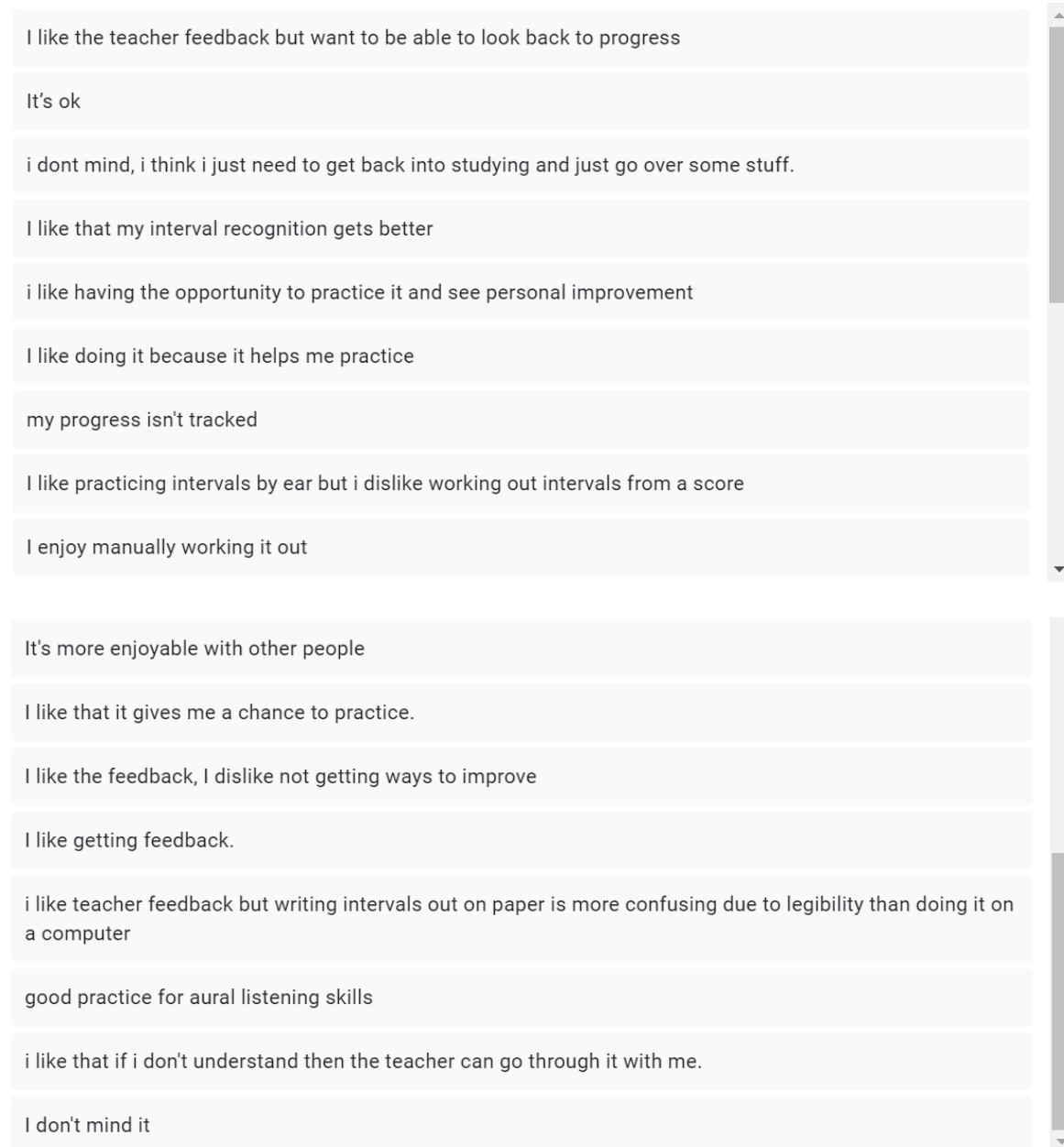
Figure 9 - Questionnaire Q6

In this question I detailed some potential features of the new system and asked students to tick the three that they would want the most. The top three I will prioritise as the main elements in the system when I come to designing and creating it.

Written Answers

What do you like/dislike about being tested on intervals in class and writing answers on paper? (e.g. I like the teacher feedback but I dislike that my progress isn't tracked)

19 responses



I like the teacher feedback but want to be able to look back to progress

It's ok

i dont mind, i think i just need to get back into studying and just go over some stuff.

I like that my interval recognition gets better

i like having the opportunity to practice it and see personal improvement

I like doing it because it helps me practice

my progress isn't tracked

I like practicing intervals by ear but i dislike working out intervals from a score

I enjoy manually working it out

It's more enjoyable with other people

I like that it gives me a chance to practice.

I like the feedback, I dislike not getting ways to improve

I like getting feedback.

i like teacher feedback but writing intervals out on paper is more confusing due to legibility than doing it on a computer

good practice for aural listening skills

i like that if i don't understand then the teacher can go through it with me.

I don't mind it

Figure 10 - Questionnaire Q7

If you have used it - what do you like/dislike about it? (e.g. I like that it tracks my individual progress but I dislike that I can't get teacher feedback)

3 responses

Idk

I have not used it yet

Figure 11 - Questionnaire Q8

Are there any other features that you would like to suggest?

19 responses

no

Multiple testing modes

Non

i dont know

I'm not sure how to use auralia?

a bit like kahoot where you can go against your classmates on an online test

the teacher can track our progress and help us improve

tests in exam format

Not that come to mind

nope

No

n/a

don't think so!

No, everything there looks like it would be good to include!

No not really

Figure 12 - Questionnaire Q9

Summary

Overall, the majority of students feel as though they need interval practise but do not know about or use the current software. The majority of students also do not feel as if the current system, both analogue and digital, tracks their progress. Even taking account for the fact that there are L6 who may not have used or heard of the software yet, the majority of the U6 students also do not use the current software. The top three things that students would prioritise in a system are: the ability to see the intervals you have misidentified the most and test yourself on those, the ability for teachers to create tests and assign them to you and see your score, and a graph that shows your progress and if you interval recognition has improved. The majority of students like the fact that in the analogue system the teacher can give feedback, they like working out intervals by ear rather than from a score and they like working with others but dislike that they cannot easily look back and see what they have got wrong previously to improve. There was no feedback on the software as so few students use it because of the lack of teacher interaction and therefore lack of incentive.

Document Capture

In the document capture I have gathered the ways in which data is stored and presented in the current systems. I will use this information later on to help with my user interface design, structure my database, and store data.

This is an example of a mark book. The data that will be stored in the system will be similar to that which is stored in a mark book e.g., name, raw score, percentage, date of test.

[illegible]

Figure 13 - Paul Clifford's Mark Book

This is the PDF file that is created by Auralia as a summary for all of the test scores of the user. It contains the username, if the data has been filtered or if it is all the data, the topic, the level difficulty and the scores and percentages for each level and then the topic and the user as a whole. It also has the date that the PDF was created and the page number.

Practise Results Detailed by Level	Report selections		
	Class:	All	Syllabus: All
	User:	192344	Topic: All
	Date range:	All	

Class: Default Class

User: 192344

Syllabus: Auralia 4 / Musition 4

Topic: Interval Recognition			
Level	Correct	Attempted	Percentage
Level 5	1	1	100.00
Level 6	7	10	70.00
Total for this topic	8.00	11.00	72.73
Average per level for this topic	4.00	5.50	85.00
Total for this syllabus	8.00	11.00	72.73
Average per level for this syllabus	4.00	5.50	85.00
Total for this user	8.00	11.00	72.73
Average per level for this user	4.00	5.50	85.00

Figure 14 - Auralia Practise Results PDF

This is a screenshot of the Auralia testing screen. This screen is displayed once you have selected the testing difficulty and the interval is played. It shows the list of intervals that can be tested and a button that can be pressed to replay the interval.

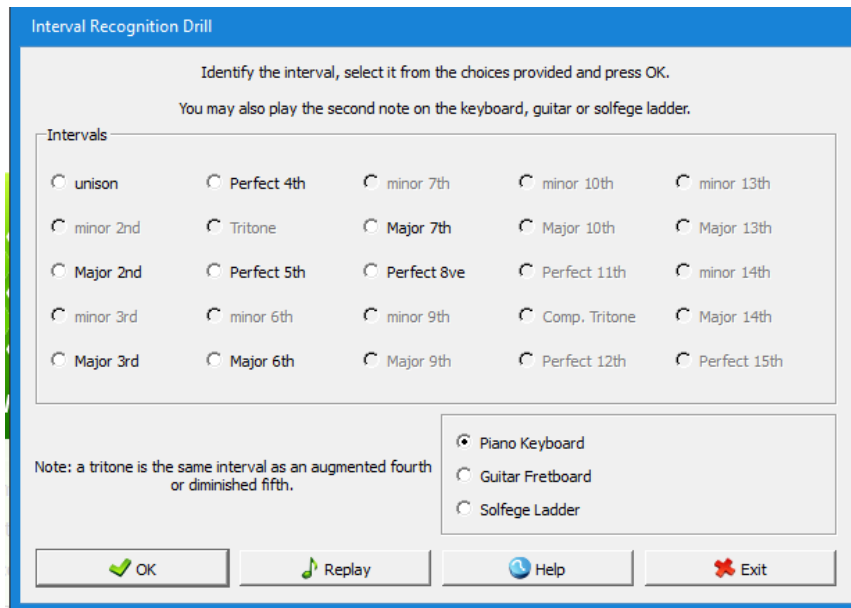


Figure 15 - Auralia Testing Menu Screenshot

This is a screenshot of the Auralia feedback screen if you answer a question correctly. It displays a tick to indicate that the question has been answered correctly and also it plays the interval again to consolidate. It also displays the scored notation of the interval and the name of the interval.

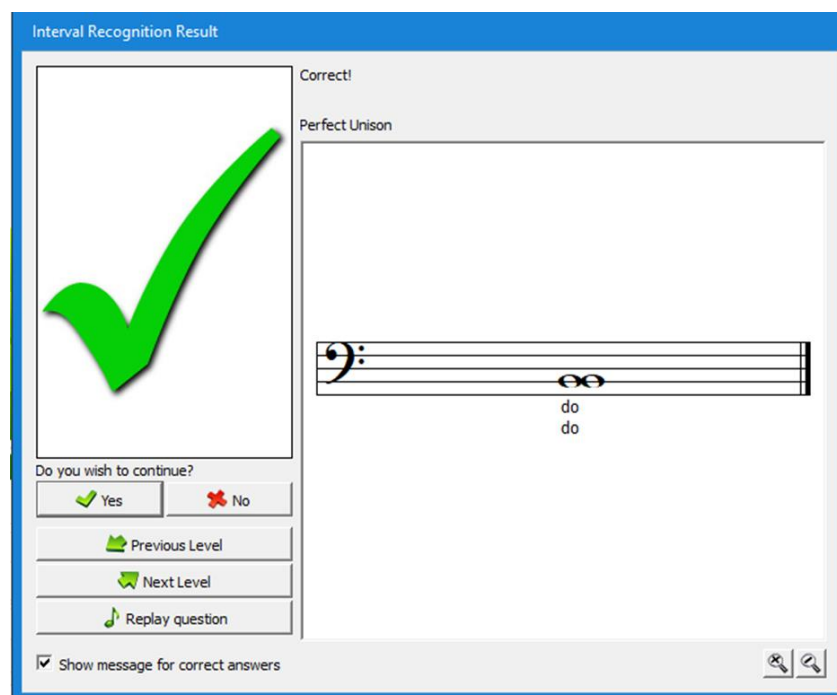


Figure 16 - Auralia Feedback Screenshot

This is a screenshot of the Auralia level select screen. This is displayed before taking a test and enables the user to select the level difficulty with level 1 being the easiest level and level 17 being the hardest. On the left is the list of levels and on the right is a summary of the intervals in the level that is being selected. At the bottom is the button to start the test.

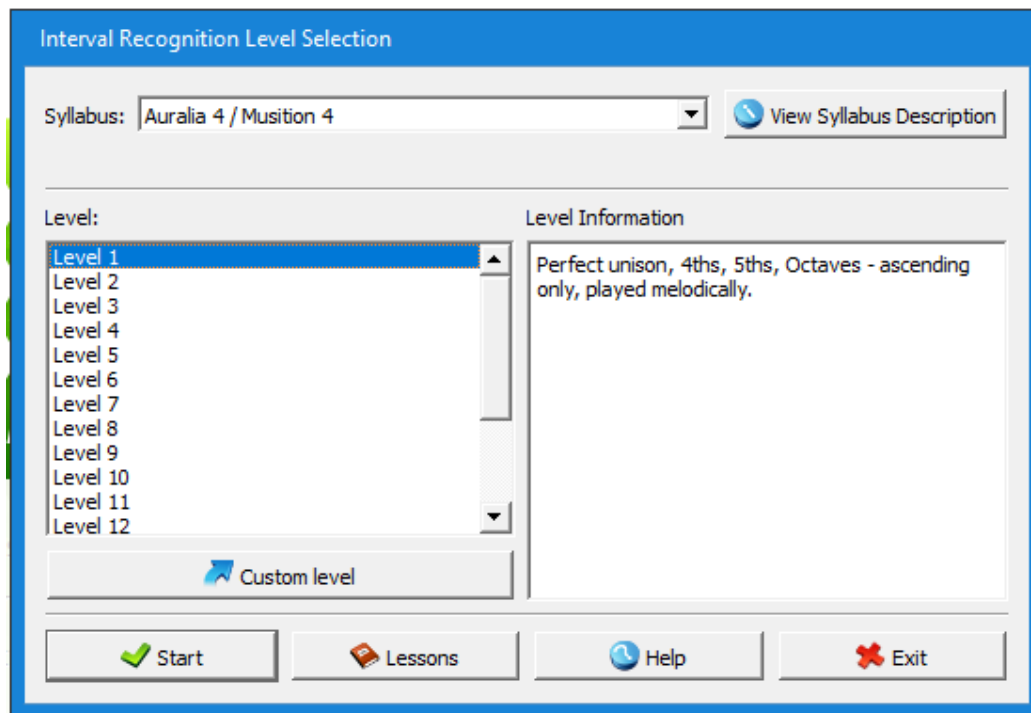


Figure 17 - Auralia Level Select Screenshot

Research Summary

In my research I have used a range of methods including an interview, questionnaire, observation, and document capture in order to understand every aspect of my problem from a range of perspectives and in a variety of mediums.

The interview has provided me with an in-depth view of the needs of my end user and gave me an opportunity to ask detailed questions and receive detailed answers. The immediate feedback from an interview rather than the delay of an email or a questionnaire meant that it was easy to discuss ideas at length and with little misunderstanding or need to re-ask or clarify questions. However, the data from an interview is difficult to display and collate.

The questionnaire has provided me with data from a large number of users of the system. The data received is easy to visualise and turn into graphs which makes it easy to interpret the overall views and desires of the users. The questionnaire questions, however, are very limited and lack detail so do not provide much in-depth detail into the needs of the users.

The observation enabled me to focus on the processes of the current systems. This allowed me to see how the current system operates in detail so that I will be able to take the key features and processes forward from the current system into my project.

The document capture has given me a good example of structures for storing data and the type of data I will need to store in my system and how it should be displayed.

Analysis

Description of Problem

After completing my research, I now have a clear understanding of the problem that I am undertaking.

At Godalming College, A level music students need to learn, and practise musical interval recognition and Paul Clifford has asked me to create a system which enables this practise. Currently, interval practise is mainly done in class. However, there is also a software that students can use to test their interval recognition but very few students use it due to the lack of teacher and peer interaction.

Therefore, I have decided to create a computerised system that tests the intervals in a similar way to the way that they are tested in class currently in order to create a more effective system that tracks the progress of the students as well as the teacher being able to set tests and access progress reports.

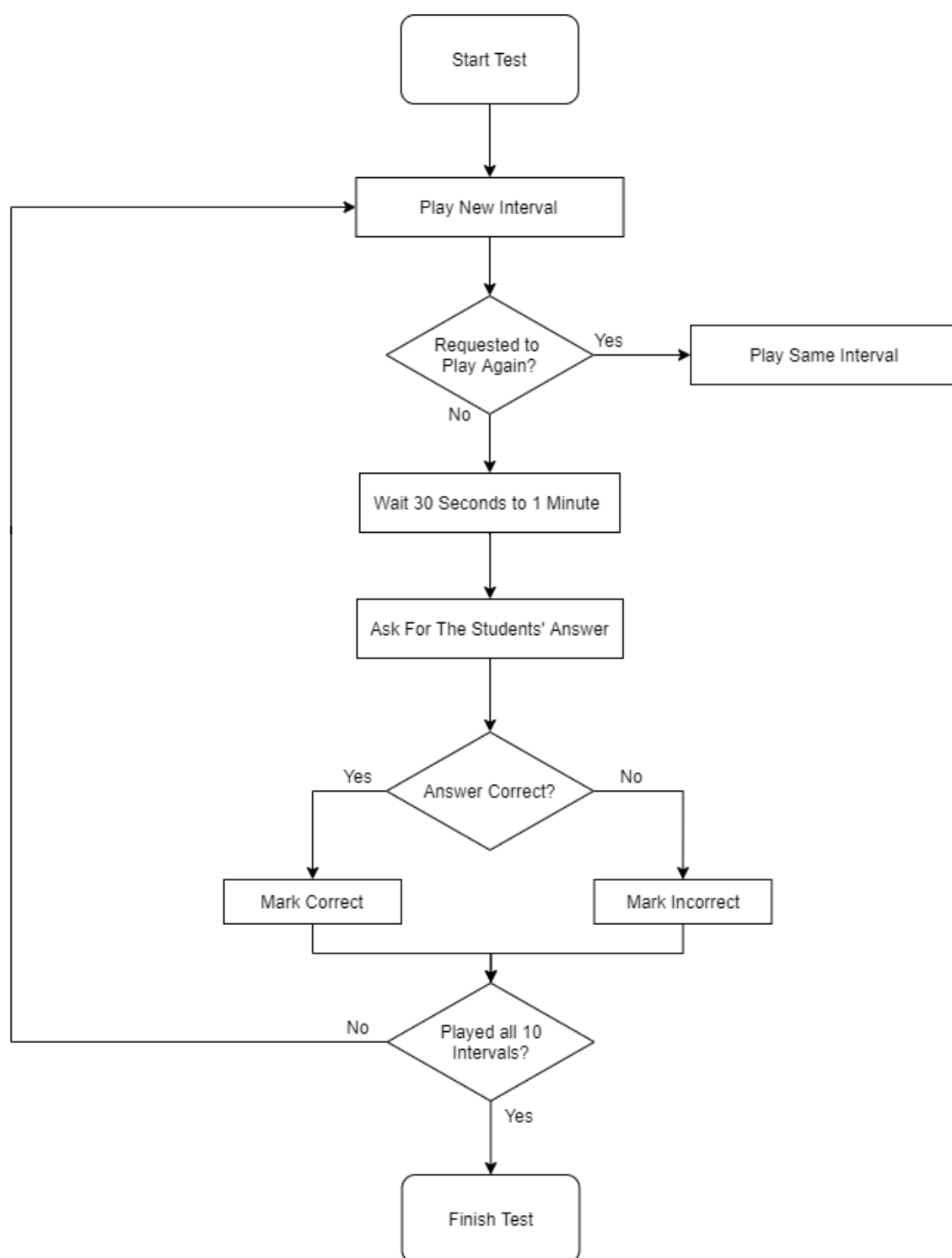
Description of Current System

Through my interview and my observation during the research I have found out how the current systems, both analogue and digital, function. I have written a condensed description and made a flow diagram and IPSO chart for these to aid my understanding of the current system.

Analogue System Description and Flow Chart

Description

- The teacher plays an interval on the piano twice.
- The teacher waits for between 30 seconds and 1 minute.
- The teacher repeats the interval if requested by a student.
- The students write down the name of the interval being played.
- The teacher goes around the class and asks what answer each student got.
- The teacher reveals the correct answer, and the students mark their answer.
- The teacher moves on to the next interval and repeats the process.
- The teacher tests 10 intervals before ending the test.

Flow Chart

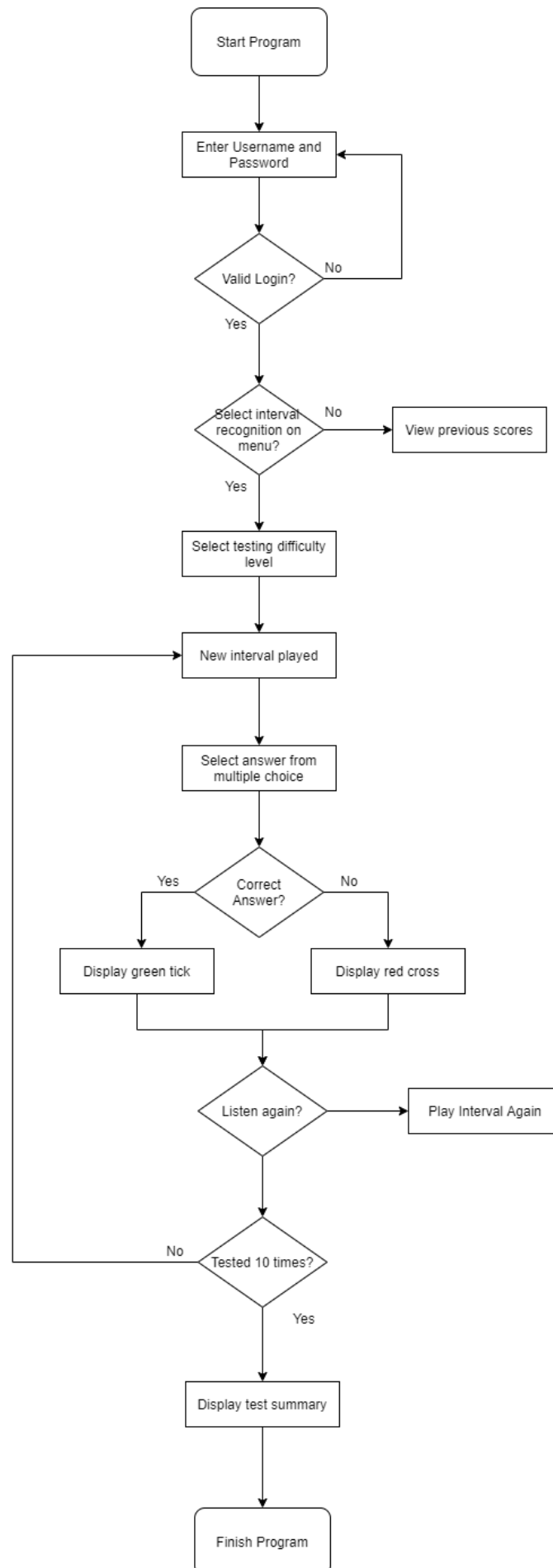
Systems Outline Chart for Current Analogue System

INPUT Guesses for the names of intervals played.	PROCESS Deciding which interval to play Play an interval. Adding up how many intervals have been played. Deciding if given answer is correct
STORAGE n/a – no answers or questions are stored.	OUTPUT Interval sound Correct answer Whether student was correct or incorrect

Digital System Description and Flow Chart

Description

- The student logs in with a username and password.
- They select the interval recognition menu option.
- The student selects the testing level.
- An interval is played.
- The student selects the interval name from a multiple choice.
- They are told if their answer is correct and are shown the musical notation of the interval they just heard and have the opportunity to listen again now that they know the answer.
- The testing loops until the student decides to stop practising.
- The screen displays your overall percentage for the test and says which intervals you have correctly and incorrectly identified.

Flow Chart

Systems Outline Chart for Current Digital System

INPUT Username and Password Difficulty level selection Multiple choice answer selection	PROCESS Randomly generate an interval that is the difficulty level selected. Determine if user input is correct.
STORAGE Names of all of the intervals Audio files of all the notes Student details and their previous scores	OUTPUT Intervals through speakers Multiple choice options on screen Tick or cross if recognition is correct. Table displaying previous scores

Systems Outline Chart for Proposed System

After taking an in depth look at the current system, I have begun to outline how my proposed system will function:

INPUT Login information (S + T) Class code (S) Difficulty level for test (S + T) Answer to question (S) Due date for assignment (T) Class Name for assignment (T) Student name when searching for test results (T)	PROCESS Create new account (S + T) Create new class code and create new class (T) Create randomly generated test (S + T) Generate interval to play (S) Create manual test (T) Generate test based off previous answers (S) Create graph of previous scores (S) Join/Leave class (S)
STORAGE Student information: name, student number, username, password, class (class code) (S) Teacher information: name, username, password (T) Individual test information: student number, test code, date of test, raw mark, percentage, incorrect intervals, (if set then whether or not the student met the deadline) (S) Summary information: percentages for every test, date of every test, average percentage (S+T) Intervals: Interval ID, interval quality, interval number, number of semitones, difficulty (S)	OUTPUT Log in page (S + T) Menu screen: create new class, set manual/generated test, view student scores (T) Menu screen: take teacher assigned test, take random test, take generated on previous answers test, view results (S) Interval sound out of speaker (S) Correct or incorrect answer message box (S) Summary page after test (S) View graph (S + T)

S = Student

T = Teacher

S + T = Student and Teacher

Prospective Users

I have identified all the users that will use my system and described how they will use it in order to develop a deeper understanding of what my system needs and how it will function for the different users.

Users		Role
Organisation	Person	
Godalming College	Music Teacher - Paul Clifford	Create and set tests to students and will track their progress in tests and practise.
	Other music A level teachers	Set tests to students if they are struggling with the interval elements on the other side of the A Level course.
Godalming College	Students	Take tests set by teachers and do interval practise with the randomly generated tests.

User Needs and Acceptable Limitations

After reviewing my research, I have come up with a list of the needs for each user as well as some limitations to the system. These are important so that when I am designing my system I can see if all of the user's needs are met as well as knowing what limits my system will have.

Teachers:

- Log in to personal account that has details of his classes and students.
- Create class groups that tests can be set to
- Set customised tests where he can choose every interval.
- Set randomised tests where intervals are randomly chosen from intervals in the chosen difficulty level.
- View student percentages on tests and a graph of their progress over time on set tests
- View student progress on their personal practise tests

Students:

- Log in to personal account with access to their previous scores and details on their previous achievement.
- Join a class group and be able to be assigned tests from a teacher.
- Take tests assigned by the teacher.
- Take randomised practise tests.
- Take practise tests that use the intervals that they get wrong the most frequently.
- View a graph of their percentages on tests over time and a list of intervals and how well they have been able to identify them.

There will be some limitations for the system:

- Due to the timeframe of the project, it will be coded in VB.net which means that the program must be downloaded onto the computers in college and will not be able to be accessed by the students from home.
- The graphs and progress tracking must start from scratch when the students use the program as no previous data about scores and the intervals tested has been saved from the in-class testing meaning it could not be saved in the system.
- The teachers and the students will need to be trained to use the new system as it will be different from the old one.

Data Sources and Destinations

I have created tables that detail, in the current and proposed systems, the data in the system and where it comes from and then where it is used. This is important as it will help me gain better understanding of the data I will need to handle when I come to create my system and where it will need to go.

Current system (Analogue)	Data	Source	Destination
	Student Name	Register	Mark book
	Raw Mark	Test paper	Mark book
	Grade	Paul Clifford	Mark book
	Percentage	Test paper	Mark book
	Date of Test	Paul Clifford	Mark book

Current System (Digital)	Data	Source	Destination
	Username	Log in	Practise results sheet
	Testing topic	Menu selection	Practise results sheet
	Difficulty level	Difficulty selection	Practise results sheet
	Raw Mark	Test	Practise results sheet
	Total number of questions	Test	Practise results sheet
	Percentage	Test	Practise results sheet
	Student Average Percentage	Test	Practise results sheet

Proposed system	Data	Source	Destination
	Difficulty level	Difficulty selection	Test generation/Feedback
	Intervals correct/incorrect	Test	Test generation/Summary of intervals
	Raw mark	Test	Tests Feedback
	Percentage	Test	Tests Feedback/Graph
Proposed system	Average Percentage	Test	Tests Feedback/Graph

Volumetrics and Data Dictionary

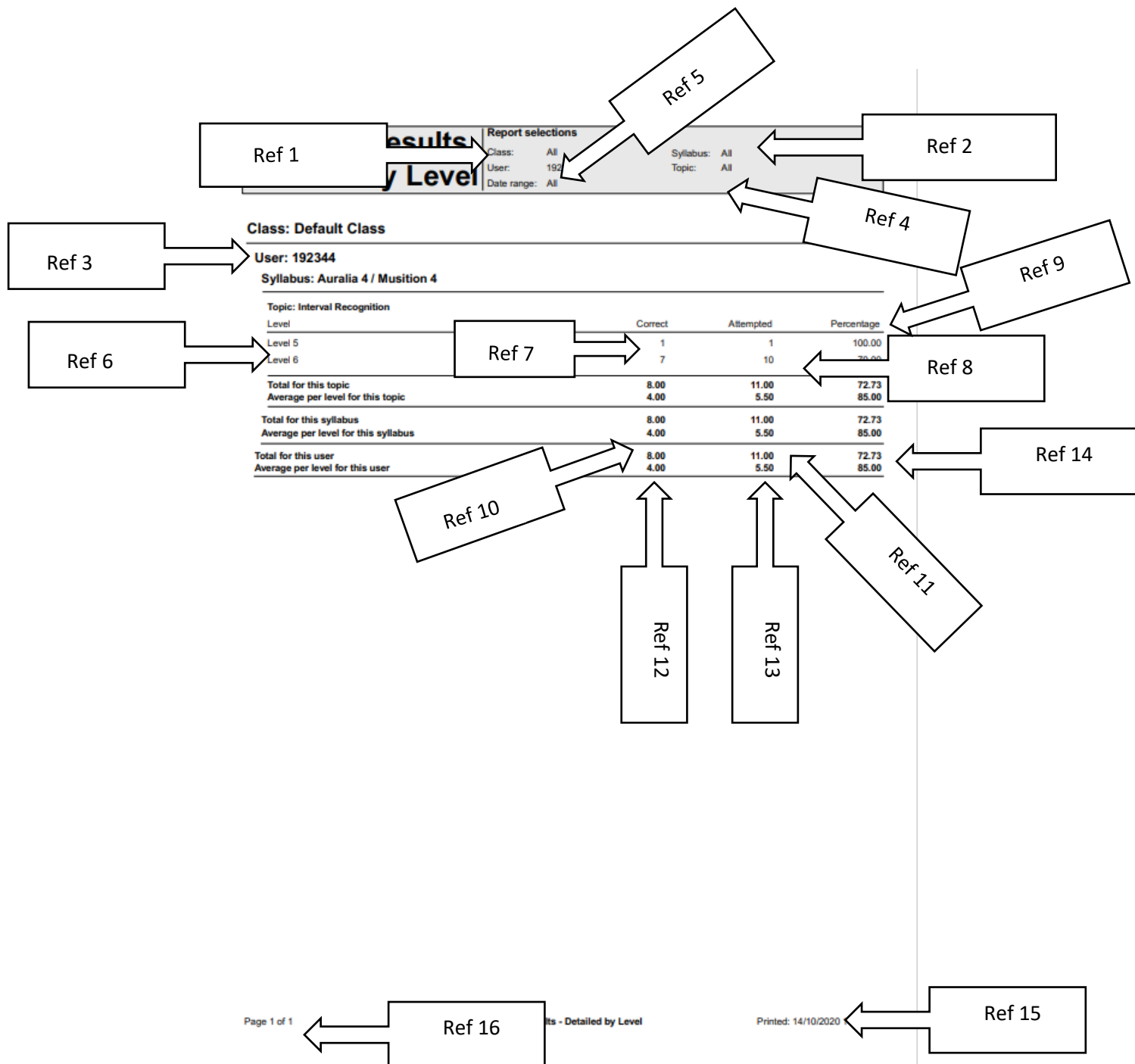
Below I have detailed the volumetrics and created a data dictionary for the two forms of data storage in the current system that I had captured in the document capture. This will give me a good idea of what data and how much of it will be needing to be stored when I come to make my system. I have also included said document capture with labels corresponding to the references in the data dictionary.

Volumetrics					
Document description	System	Document	Name	Sheet	
Paul's Mark Book	Music Grades	Figure 13	Katie Taylor	1 of 1	
Stationery ref.	Size	Number of parts	Method of preparation		
	A4	1	Handwritten		
Filing sequence	Medium		Prepared by		
Chronological	Paper Ring binder		Paul Clifford		
Frequency of preparation	Retention period		Location of file		
Once per term	2 years		Music office		
Volume	Minimum	Maximum	Av/Abs	Growth rate/fluctuations	
	4	12	8	4 – 12 exams per year that need to be documented in music	
Users/receipts	Purpose		Frequency of use		
Music Teachers	To keep track of the scores of music students in tests		Once per test		
Data Dictionary					
Ref	Name	Data Type	Regex	Occurrence	Source of data / description
1	Student Name	String	([A-Z][a-zA-Z]*)	x14	Register
2	Raw Mark	Integer	^\d+\$	x98	Test paper
3	Percentage	Integer	^[0-9]\$\^[1-9][0-9]\$\^(100)\$	x42	Test paper
4	Grade	Char	(^[A-F])(?<=A)*?\$)	x156	Paul Clifford
5	Date of Test	Date	^\d{1,2}\d{1,2}	x3	Paul Clifford

The image shows a handwritten mark book page with columns for student names, marks, and percentages. Annotations with arrows point to specific data points corresponding to the references in the Data Dictionary:

- Ref 1 (Blocked for data protection)**: Points to the student name 'C. (C)'.
- Ref 2**: Points to the raw mark '41'.
- Ref 3**: Points to the percentage '57'.
- Ref 4**: Points to the grade 'B'.
- Ref 5**: Points to the date '14/10'.

Volumetrics						
Document description		System		Document	Name	Sheet
Practise Results PDF		Auralia		Figure 14	Katie Taylor	1 of 1
Stationery ref.		Size		Number of parts	Method of preparation	
		A4		1	Generated by Auralia	
Filing sequence			Medium		Prepared by	
Chronologically			Digital		Auralia	
Frequency of preparation			Retention period		Location of file	
Whenever user clicks to generate a report			As long as the user wants to keep it		Auralia database	
Volume	Minimum	Maximum	Av/Abs			Growth rate/fluctuations
	0	10	5			The amount of data on each review will increase with each test taken
Users/receipts		Purpose				Frequency of use
Katie Taylor		To review data on the practise questions that have been taken				X1 per test
Data Dictionary						
Ref	Name	Data Type	Regex		Occurrence	Source of data / description
1	Class	String	[A-Za-z]+		X1	Menu
2	Syllabus	String	[A-Za-z]+		X1	Menu
3	Username	String	[A-Za-z]+		X1	Log in
4	Topic	String	[A-Za-z]+		X1	Menu
5	Date Range	Date	^(0?[1-9] [12][0-9] 3[01])([\\-])(0?[1-9] 1[012])([\\-])\\d{4}\$		X1	Menu
6	Level	String	Level [0-9]+		X1	Menu
7	Correct	Integer	^\\d+\$		X2	Test
8	Attempted	Integer	^\\d+\$		X2	Test
9	Percentage	Integer	^[0-9]\$^[1-9][0-9]\$^(100)\$		X2	Test
10	Total Correct	Integer	^\\d+\$		X3	Test
11	Total Attempted	Integer	^\\d+\$		X3	Test
12	Average Correct	Decimal	\\d+(\\.\\d{1,2})?		X3	Test
13	Average Attempted	Decimal	\\d+(\\.\\d{1,2})?		X3	Test
14	Average Percentage	Decimal	^(?:100(?:\\.00?)? \\d?\\d(?:\\.\\d\\d?)?)\$		X3	Test
15	Date Printed	Date	^(0?[1-9] [12][0-9] 3[01])([\\-])(0?[1-9] 1[012])([\\-])\\d{4}\$		X1	Computer
16	Page Number	String	Page [0-9] of [0-9]		X1	Computer

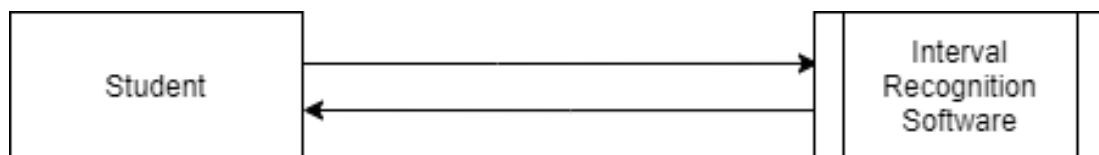


Data Flow Diagrams

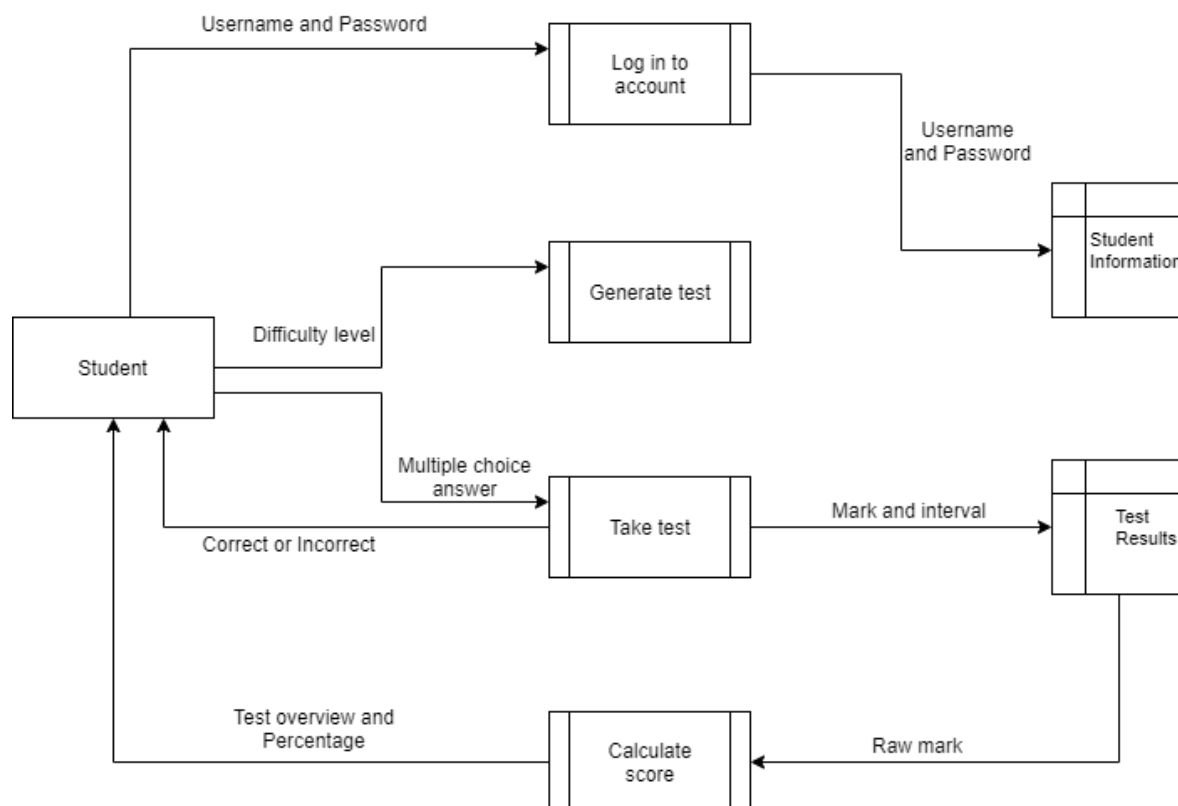
I have created level 0 and level 1 data flow diagrams for the existing system and the proposed system. These will give me an idea of the overview of how data will travel in the system and a more detailed view of how data will travel. This will enable me to effectively design my system, so the data moves as intended.

Existing System

Level 0

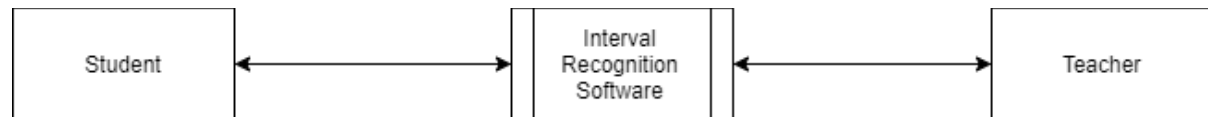


Level 1

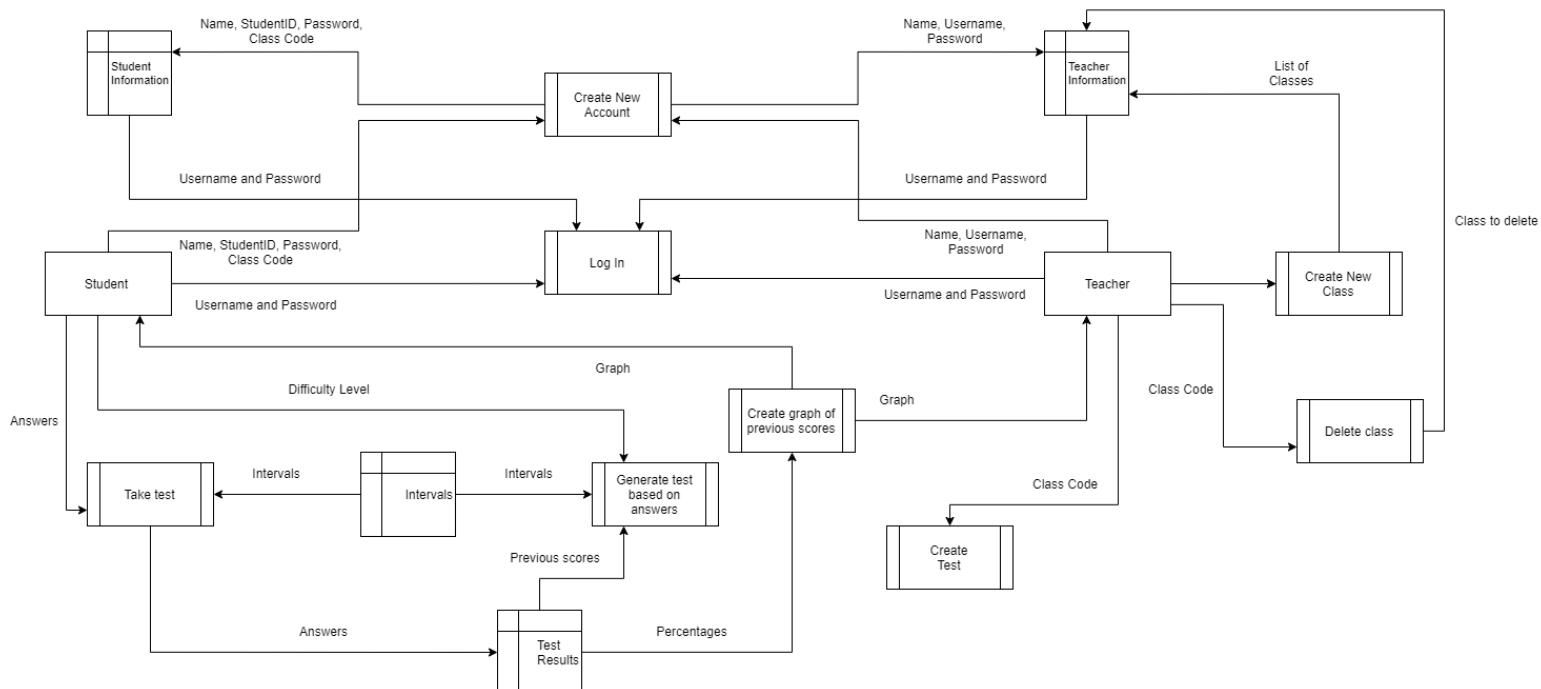


Proposed System

Level 0



Level 1



Requirements

Following my analysis I have now assessed what I will need in my system in order for it to function effectively. Below I have detailed a list of requirements that will guide my design and implementation process:

1. The program should be server based.
 - 1.1. Multiple clients could be able to connect.
 - 1.2. The clients must be able to connect simultaneously.
2. The program must be easy and efficient to use.
 - 2.1. The navigation tools (e.g., buttons) must be labelled descriptively.
 - 2.2. The program interface must be laid out simply and clearly.
 - 2.3. There must be a simple and cohesive colour scheme.
3. Students and teachers must be able to create accounts.
 - 3.1. When a student creates an account, the following data must be captured: student number, first name, surname, and password.
 - 3.2. When a teacher creates an account, the following data must be captured: teacher initials, first name, surname, and password.
 - 3.3. This data must be added to the database.
4. Students and teachers must log in when they open the program.
 - 4.1. They must type in a username and password into textboxes.
 - 4.2. The data entered is checked against the database.
 - 4.3. If the username and password match a record, then the program is opened.
5. Teachers can create classes.
 - 5.1. The teacher is prompted to enter a class name e.g., "2020-2021".
 - 5.2. A random code is generated which is the class code which can be given to students.
 - 5.3. The class code is stored in the database and associated with the teacher who runs the class and the students who have entered the class.
6. Teachers can set tests to the class.
 - 6.1. The teacher can select either a randomly generated test or a manual test.
 - 6.1.1. For the randomly generated test the teacher chooses a difficulty level
 - 6.1.2. An algorithm then determines the 10 intervals that will be tested based on the difficulty level.
 - 6.1.3. For the manual test, the teacher will be able to select the 10 intervals tested from a list of all intervals.
 - 6.2. The teacher will select which class the test will be set to.
 - 6.3. The teacher will select a due date for the test.
7. Random intervals can be generated.
 - 7.1. If a random test is taken by a student an algorithm will determine what interval is played based on difficulty level and what note it will start on.
 - 7.2. A random number index will be taken from the array of notes which will be the starting note.
 - 7.3. A direction is then chosen randomly chosen which will indicate what the second note of the interval will be.

8. Intervals are fetched and played.
 - 8.1. The two notes that are required to play the interval that has been selected or generated are found.
 - 8.2. It is played out of the computer's speaker using the console sound system.
9. Each test will run as follows:
 - 9.1. For each of the 10 intervals the system will
 - 9.1.1. The system will fetch the interval.
 - 9.1.2. The system will play the interval as in requirement 8.
 - 9.1.3. The student can select to hear the interval again.
 - 9.1.4. The student will select the answer from a multiple-choice list.
 - 9.1.5. Once the answer has been selected the system will display if the answer was correct or incorrect.
 - 9.2. The data from the test (interval name, if correctly identified, percentage, date) is saved to the database.
 - 9.3. The student will be able to view an overview page and are given a percentage grade.
10. The students can take the tests set by the teachers.
 - 10.1. They can select the assignment they want to take from a list of assignments from teachers.
 - 10.2. The test will run as in requirement 9.
11. The students can create and take a randomly generated test.
 - 11.1. The student will select a difficulty level.
 - 11.2. The test will run as in requirement 9.
12. The student can create a test based on their previous test results.
 - 12.1. An algorithm will determine the 10 intervals that the students have misidentified the most in the most recent previous tests.
 - 12.2. The test will run as in requirement 9.
13. A graph of the student's progress can be created and viewed.
 - 13.1. The percentages from all previous tests and the date that they were taken is plotted in a graph.
 - 13.2. A button can be pressed to set the graph to display the average score per pay and date plotted on the graph.
 - 13.3. The graph can be viewed by students by clicking on their results page.
 - 13.4. The graph can be viewed by teachers by clicking on the student pages and selecting the student.

Requirements Dialogue

1. I have chosen to do the project as a database project, due to the fact that the research in the interview and the questionnaire highlighted that the main issue with the previous system was that there was no student-teacher interaction, and that one of the top three things wanted from the new system by the students was the element that teachers should be able to see results and give feedback which was a sentiment echoed by Paul Clifford in his interview. Therefore, I have decided to use a database as it allows the data from the students' tests to be added to the database and then accessed by the teachers. I have considered doing the program as a web-based application however as Paul Clifford stated in the interview that they would only be using the software on the college music department

computers it made sense to make it a downloadable application rather than web based. I have instead decided to code it in VB.net as this is where my skills are currently and will therefore make the development of the software more streamlined.

2. The program must be easy and efficient to use due to the fact that Godalming College has a quick turnaround of students as they only stay at college for two years. This means that the teachers will not have time to teach the software to the students every year with each new intake if the program is overly complicated. Therefore, the program should be intuitive to use and not require training.
3. The students and teachers must be able to create accounts. For the students this is because in my research and the questionnaire the number one thing wanted from the system from the students would be to be able to track their personal progress and view the intervals that they have misidentified in the past. Therefore, it makes sense for each student to have an individual account, so their tests and their progress is assigned to their personal account. For the teachers this is because they will have different capabilities than a student account and need special priorities such as being able to see all student results.
4. Students and teachers must log in. This is to authenticate the user, so the student or teacher is using the correct account and also prevents unauthorised access. Using the correct account to do the tests is important as if a student used the wrong account then it would invalidate the data about their interval recognition and make the algorithm to create personalised tests, which is one of the main desires for the students, less effective.
5. Teachers can set classes. This is so that the classes in college can be easily grouped together in the system and means that when the teacher comes to setting the tests, he would not have to manually select each student in the class.
6. The teacher can set a manual or randomly generated test to a class. The ability to set a manual test is because in the interview Paul Clifford expressed a desire to be able to manually select intervals in a test as he might want to test them on specific intervals that are relevant to another part of the course or maybe just ones he knows are typically challenging. There is also an option to set a randomly generated test so that Paul does not have to spend the time manually creating a test but still wants to set one.
7. Random intervals can be generated by using a random number generator to select the starting note from an array of notes and then a random number generator to select the second note from the array of notes which will then loop until the interval is one that is valid for that difficulty level. I considered other ways of doing this such as having an 2D array of intervals however this is more complicated as I would then have to have every single combination of notes which would be a lot of time to hard code as I would have to include ascending and descending intervals.
8. The interval sounds are played from the console . I considered creating my own sound files for each note however this would be very time consuming and also take up a lot of storage as I would need to download the sound files onto every computer in the computer room. I

therefore decided to use the console sound, as Paul Clifford had stated in his interview that he did not mind a generic sound and did not want an array of instrument sounds.

9. When the student takes the test, it will run by playing the interval and then the student being able to select the interval name from a multiple-choice list. This is because in his interview Paul Clifford stated that he would like the option selection to be multiple choice for the quality and number separately. I had considered using textboxes as a form of input however this is less efficient than multiple choice and will require lots of validation as the answer could be inputted in multiple formats in a textbox.
10. The students can take tests set by teachers. This is important as it creates the element of the student teacher interaction that was wanted by both the teachers and the students from my research. It enables students to practise and learn the intervals that the teacher wants them to which is very similar to the current analogue system in place.
11. A student should be able to take a randomly generated test with a difficulty level as specified by Paul in his interview. This is important as it is the best way for students to practise a wide range of intervals and then once they have taken a few random tests they can then take the test based on their previous scores.
12. The students can create a test based on their previous scores. In the questionnaire the students said that the most important feature they would want in the program is the ability to track the intervals they have guessed incorrectly. Therefore, this testing feature is useful as it puts this data into a useful feature that can help the students master intervals they may be struggling with.
13. A graph is created of the student percentage scores over time. I considered the student overview page having a breakdown of every interval and how it has been identified however that would be needlessly complex and in Paul's interview he stated that he wanted a simple way to quickly view data and easily be able to tell if a student has been improving or not. The easiest way to do this is to do a graph of percentage scores over time as this is a simple and visual way of displaying the information. The students also wanted to have visual confirmation of their progress over time to motivate them to continue practising which is provided by a graph.

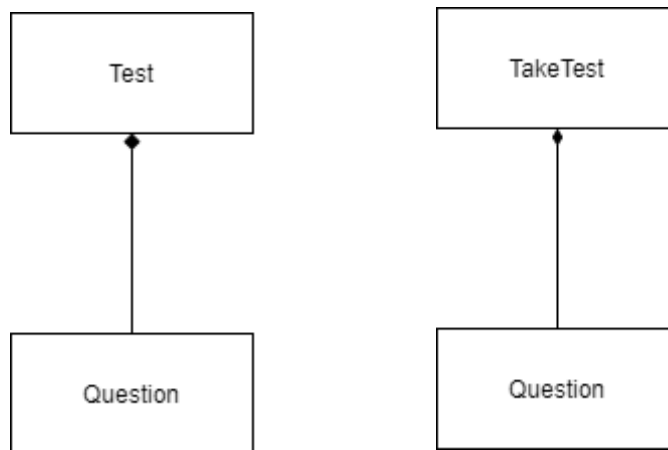
Analysis Summary

In summary, in the analysis I have analysed my research and the current systems using formal techniques such as flow diagrams, data flow diagrams and data dictionaries in order to gain a better understanding of the systems and I have used this analysis to begin to analyse and think about my proposed system. From this, I have been able to come up with a comprehensive and detailed list of requirements for my system which I will use throughout the project to refer back to in order to see what needs to be developed in order to meet the user needs and the requirements.

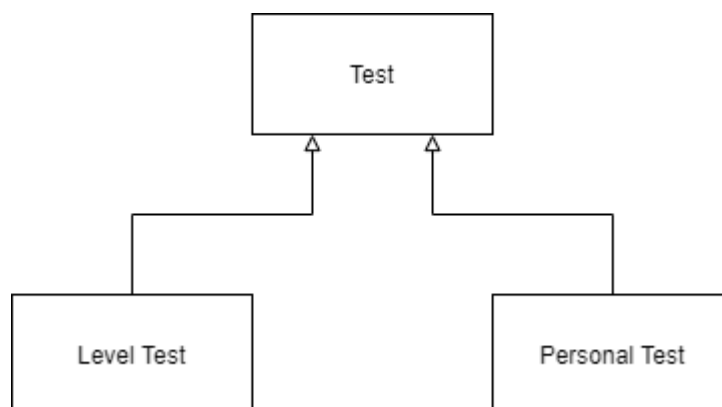
Design

Object Analysis Diagrams

As I will be coding the solution with object-oriented programming, I have created simple object analysis diagrams for the proposed system including elements of inheritance and composition. This is for me to get some idea as to how I will be structuring my code for the system.



The object analysis diagram above shows how the classes Test and TakeTest will both be composed of the class Question.

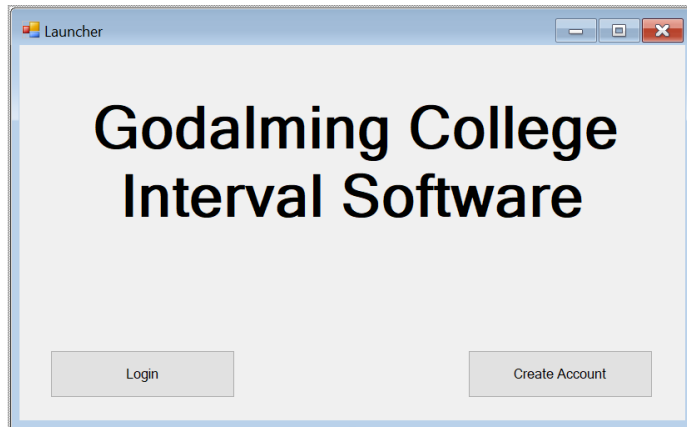


The object analysis diagram shows how the classes level test and personal test will be inherited from the class test.

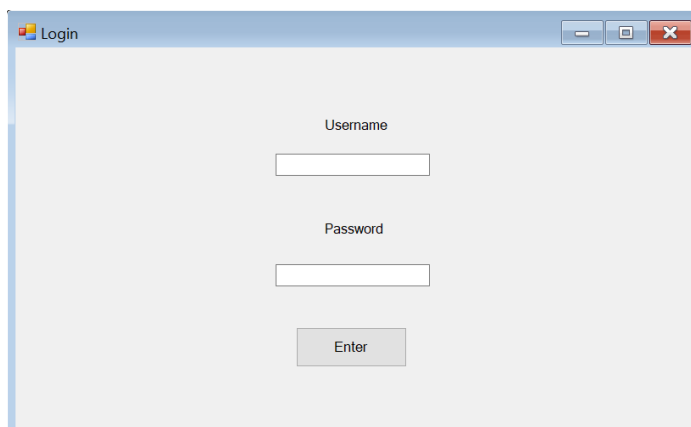
My system will also have a class for the database connection and classes for each form used.

User Interface

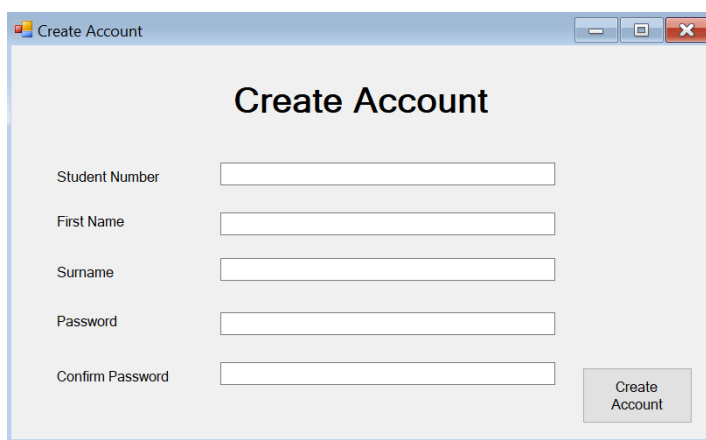
I have designed forms for my user interface for my program so that when I come to create my program I have a clear idea of what I will need in my program and how it will be displayed.



This will be the interface when the program is launched. It has a large title for the program, and it has two buttons: one labelled "login" and one labelled "create account". If the user were to click the login button it will take them to the login page and if they click the create account button it will take them to the create account page.



This is the interface for the log in page. There are two textboxes in which the user will be able to enter their username and password. Once they have entered their username and password, they can click the enter button. If the login details are valid then it will take the user to either the student or teacher menus, depending on what user is logging in.



This is the interface for creating an account. There are 5 textboxes for the user to enter the relevant information including student number (or TeacherID for the teacher create account page), name and password. Once they have entered their information and it has been validated, they will click the create account button which will add the information to the database and take them to the log in page.

Student Menu

Account Assignments Manual Test Automatic Test Results

Username [data]

Name [data]

Account Type [data]

Class [data]

Teacher [data]

This is the first tab of the student menu. It displays the basic account information such as the username, name, account type, class and teacher of the student that has logged in. I have created this page so that the user can check that they are the user that is currently logged into the program and can view their account details easily.

Student Menu

Account Assignments Manual Test Automatic Test Results

Assignments

Test Name	Teacher	Due Date
Link To Test	[data]	[data]

Completed Assignments

Test Name	Teacher	Date Completed	Score
[data]	[data]	[data]	[data]

This is the second tab of the student menu. It has a list of the current assignments and the completed assignments which are tests that have been set by the teacher. The assignments section will include the test name (which will be a link to click to access the assignment), the teacher that set the test and the due date. The completed assignments section will include the test name, teacher, date completed and the score the student achieved on the test.

Student Menu

Account Assignments Manual Test Automatic Test Results

Level 1 All intervals - only need to identify the number

Level 2 Major, minor and perfect intervals

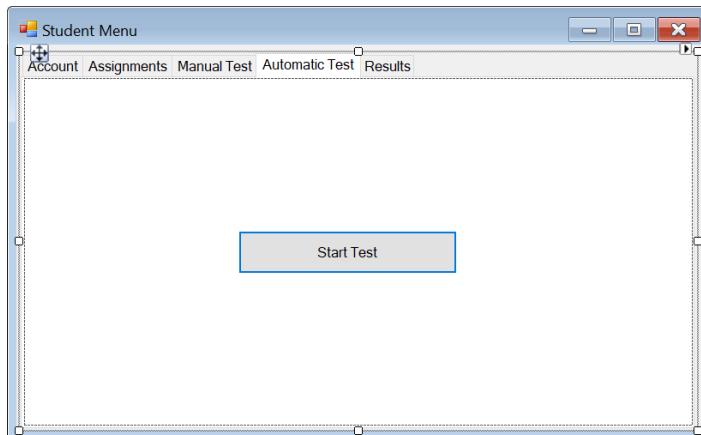
Level 3 Major, minor, perfect, augmented and diminished intervals

Level 4 Major, minor, perfect, augmented, diminished and compound intervals

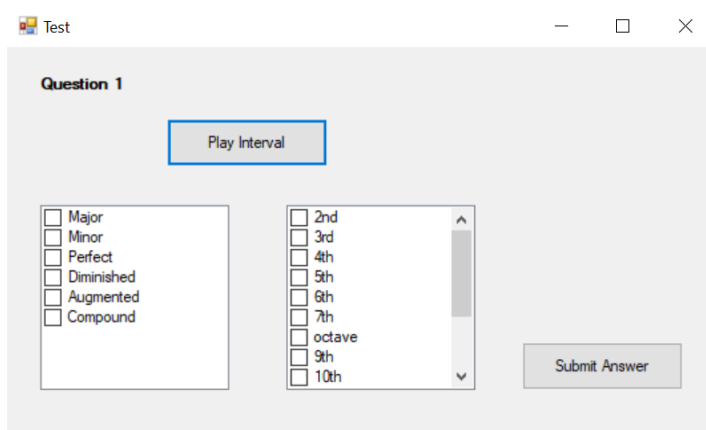
Level 1
Level 2
Level 3
Level 4

Start Test

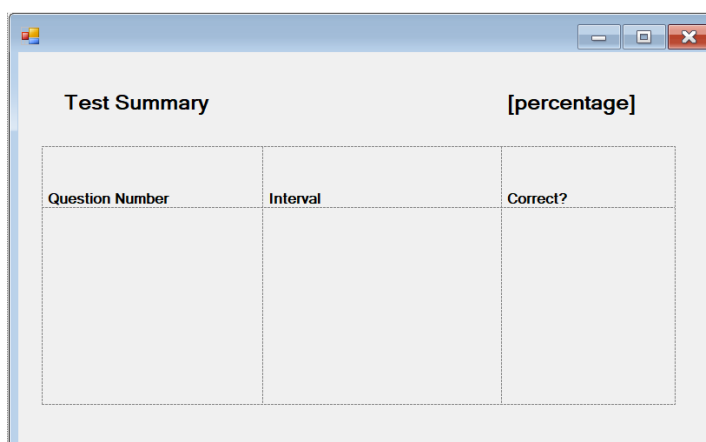
This is the third tab of the student menu. It has a description of the four levels of tests available to take as well as a drop-down list of the four levels so that the student can select the level they want to test. There is then a button to start the test once the student has selected the level. This button will then open the test.



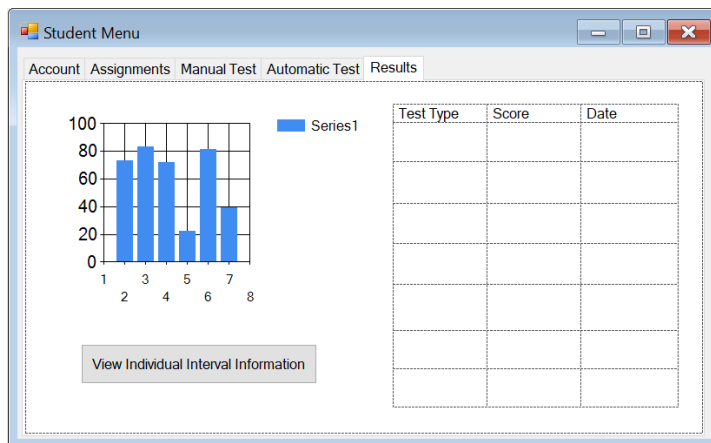
This is the fourth tab in the student menu which is the automatic test. Due to the fact that the test is generated from previous results there is no need for the student to select anything before starting the test and therefore the only thing on the page is a button which will generate the test and take the student to the test.



This is the design for the user interface for the tests – it will be the same for both manual and automatic tests. At the top will be the question number and then below is a button that when pressed will play the interval. There are then two checkbox lists from which the student can select the quality and then the number. Once they have selected their guess, they will then click the submit answer button which will take them to the next question.



This is the test summary screen which will be displayed once the student has submitted their answer on the final question. It displays their percentage score in the top right and then a table containing the question number, the interval and then if they had identified the interval correctly. Once they close this screen it will take the student back to the main menu.



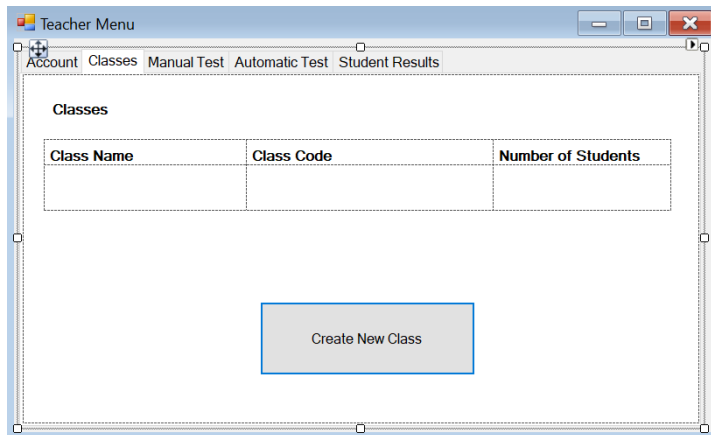
This is the fifth tab in the student menu which has an overview of the results of the student that has logged in. It will have a graph plotting the percentage score over time and then a table next to it displaying the score and the date of the test. There will also be an option to group tests by average per date on the graph. There is also a button leading to a detailed breakdown of their performance on each interval.

Interval Name	Times Tested	Times Correct	% Recognition

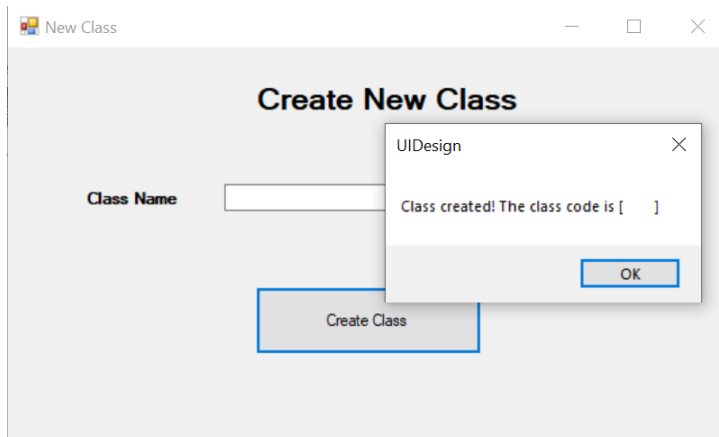
This is the detailed breakdown of the student's performance on each interval. It is a table containing the interval name, the number of times the student has been tested on it, the number of times they have identified it correctly and then the percentage of the times that they have recognised the interval correctly. When the window is closed it will take the student back to the student menu.

Username	[data]
Name	[data]
Account Type	[data]

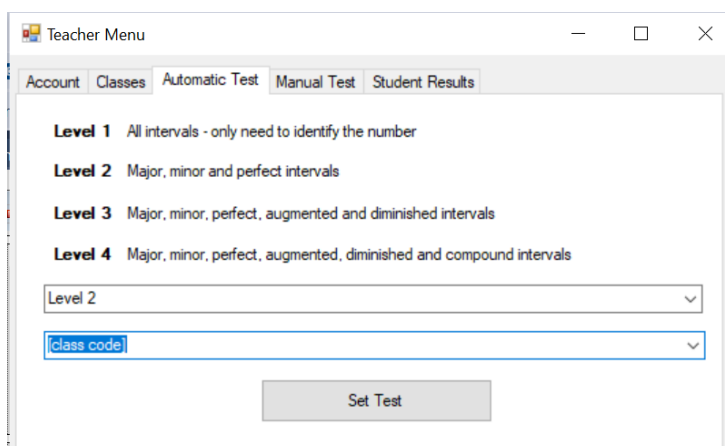
This is the first tab of the teacher menu which is what will be displayed when a teacher logs into the system. The account tab displays the account information so that the user can check that they are logged into the correct account.



This is the second tab in the teacher menu. It has a table that displays all of the teacher's classes and displays the class name, the class code and the number of students in the class. There is then a button at the bottom to create a new class. If the button is pressed, then it opens a new window to create a new class.



This window enables the teacher to create a new class. The teacher will enter the class name into the text box and then click the button at the bottom of the page. This button will then create a new class with that name and pop up a dialogue box confirming the class has been created and giving the class code. Once the teacher clicks "Ok" it will take them back to the teacher menu.



This is the third tab in the teacher menu which enables the teacher to set an automatic test. It has text describing the four levels and then two dropdowns – one to select the level and another to select the class that the teacher wants to set the test to. Once these have been selected the teacher can click the "set test" button which will set the test to the students and the test will appear in their assignments.

Teacher Menu

Account Classes Manual Test Automatic Test Student Results

Select 10 Intervals To Test

- ☒ Minor 2nd
- ☐ Major 2nd
- ☐ Minor 3rd
- ☐ Major 3rd
- ☒ Perfect 4th
- ☒ Tritone
- ☒ Perfect 5th
- ☐ Minor 6th
- ☒ Major 6th
- ☐ Minor 7th
- ☒ Major 7th
- ☐ Octave
- ☐ Minor 9th

Select Class

Select Due Date

15 November 2020

Set Test

This is the fourth tab of the teacher menu. There is a checkbox list on the left which will allow the teacher to select any 10 intervals to be tested. There is then a dropdown list of all of the classes so the teacher can select which class to set the assignment to, and then a date selector which will allow them to select the due date of the test. They can then click the “set test” button which will set the test as an assignment to the students in that class.

Teacher Menu

Account Classes Manual Test Automatic Test Student Results

Student Data

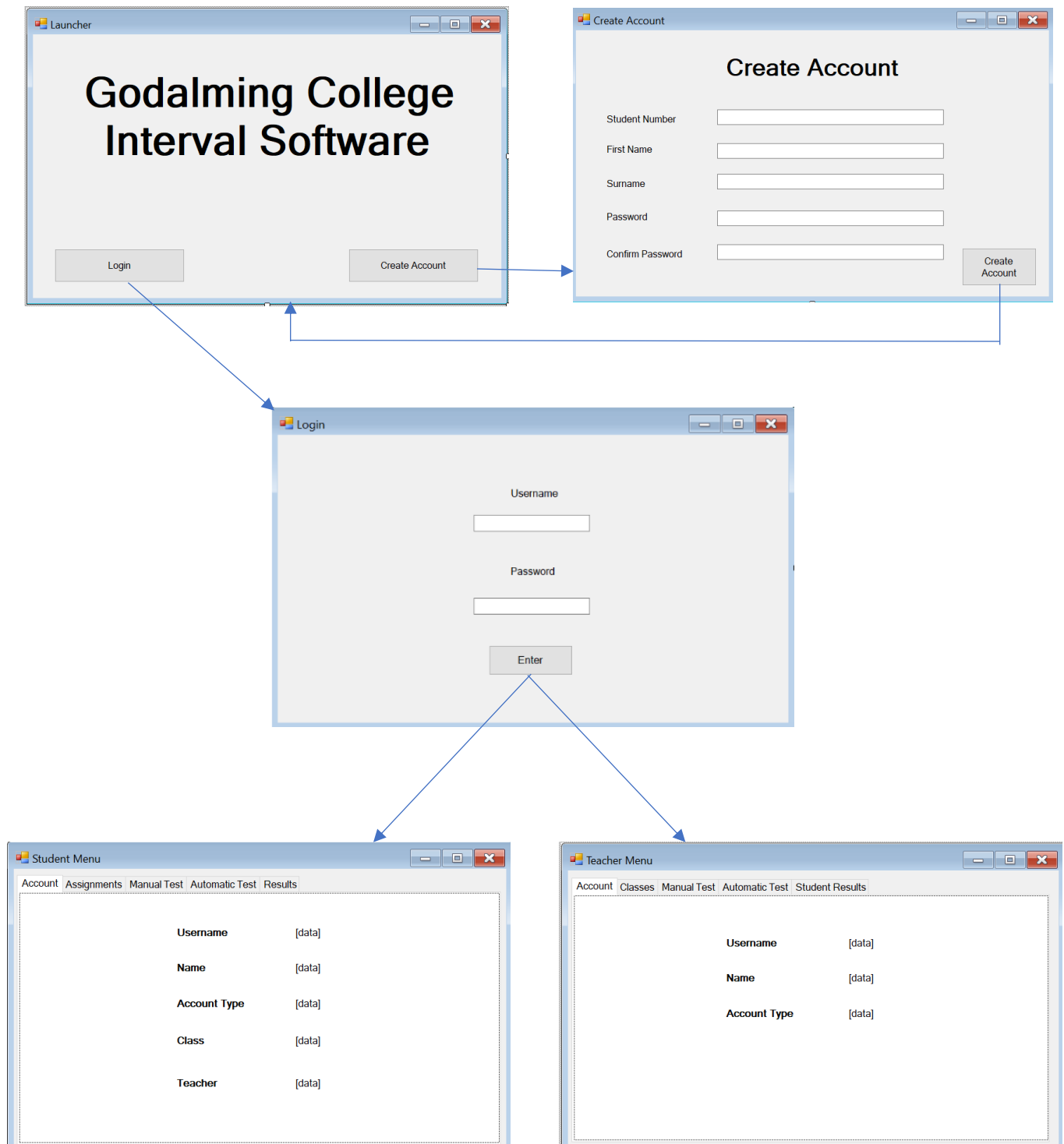
Name	Class	Average Score	View Graph

This is the fifth tab in the teacher menu which displays the student data. In the table it will display the name of the student, what class they are in, their average score on tests and then will have a link that the teacher can click which will display a graph of the percentage scores of the students over time.

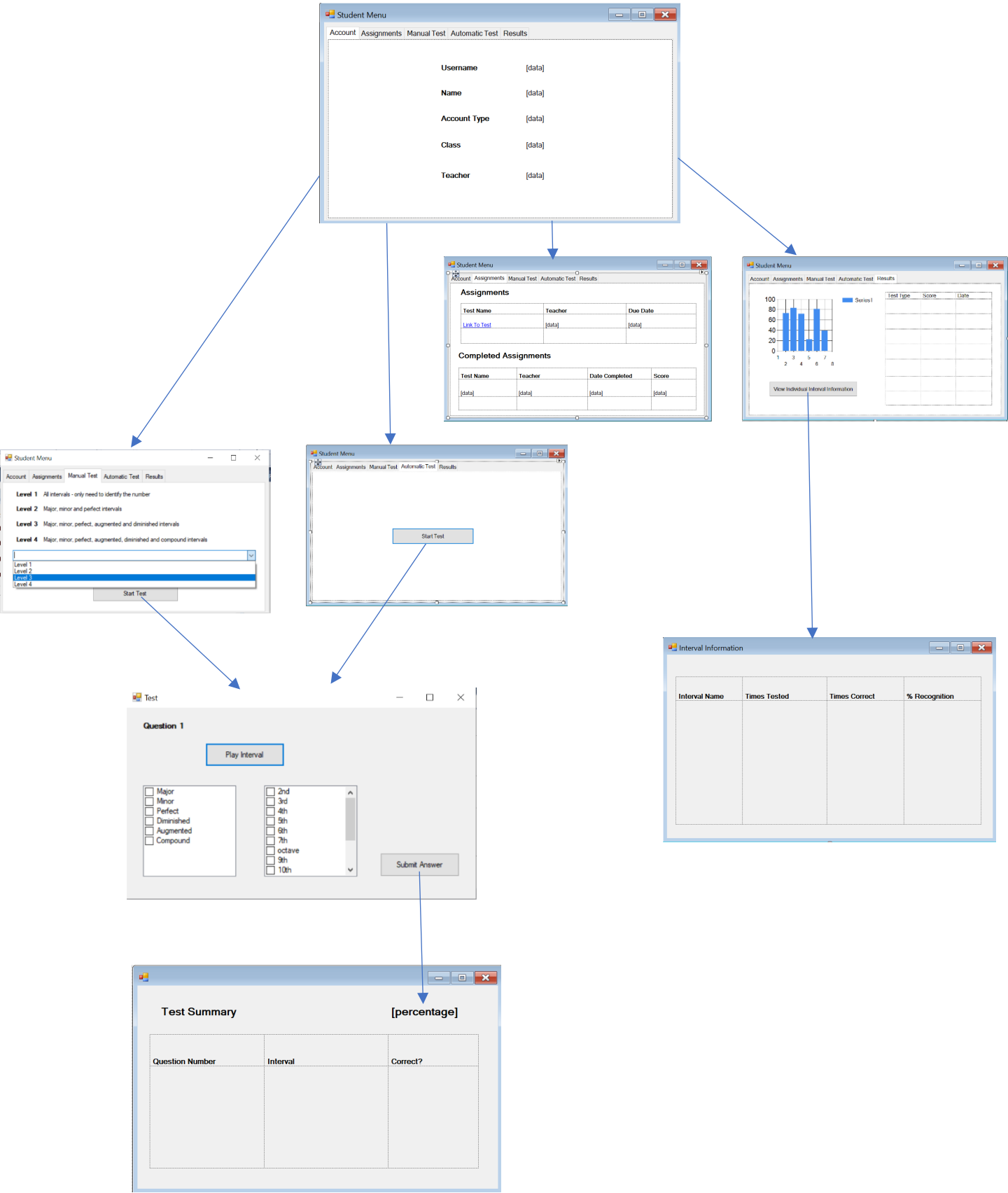
Navigation Diagrams

I have created some navigation diagrams for the user interface designs for my system. These demonstrate how the system will be navigated and how the user will be able to get from one screen to the next. This is useful as it clearly demonstrates how the system will function and will be navigated which will make it much easier to program as I am aware of what each button and form should do and where it should take the user.

Main Menus



Student Menu



Teacher Menu



Data Structures/Storage

Database Normalisation

From the IPSO chart which I created from my proposed system in the analysis I have come up with a list of the data that will need to be stored in my database. This data will need to be fully normalised in order to reduce inconsistencies in the database and make the database more efficient. Below I have detailed how I have normalised the data from first normal form into third normal form.

Not Normalised

Student(StudentID, Name, Password, Class, Teacher)

Teacher(TeacherID, Name, Password, Class)

Test(TestID, QuestionNumber, TestType, Interval, StartingNote, Ascending, Class, DueDate)

Result(TestID, StudentID, QuestionNumber, Correct, TotalScore, DateCompleted)

First Normal Form

I have removed the repeating attributes so that all the data is atomic within the tables.

Student(StudentID, FirstName, Surname, Password, Class, Teacher)

Teacher(TeacherID, FirstName, Surname, Password, Class)

Test(TestID, QuestionNumber, Level, Quality, Number, Semitones, StartingNote, Ascending, Class, DueDate, DateCompleted)

Result(TestID, StudentID, QuestionNumber, Correct, TotalQuestions, TotalCorrect, AverageScore, DateCompleted)

Second Normal Form

I have removed all non-key dependencies.

Student(StudentID, FirstName, Surname, Password, ClassID)

Teacher(TeacherID, FirstName, Surname, Password)

Class(ClassID, ClassName, TeacherID)

Question(TestID, QuestionNumber, IntervalID, StartingNote, Ascending, ClassID, DueDate, DateCompleted)

Interval(IntervalID, Level, Quality, Number, Semitones)

Result(TestID, StudentID, QuestionNumber, Correct, TotalQuestions, TotalCorrect, AverageScore, DateCompleted)

Assignment(TestID, TestName, ClassID, DueDate)

Third Normal Form

I have removed all partial-key dependencies.

Student(StudentID, FirstName, Surname, Password, ClassID)

Teacher(TeacherID, FirstName, Surname, Password)

Class(ClassID, ClassName, TeacherID)

Interval(IntervalID, Level, Quality, Number, Semitones)

Question(TestID, QuestionNumber, IntervalID, StartingNote, Direction)

Assignment(TestID, TestName, ClassID, DueDate)

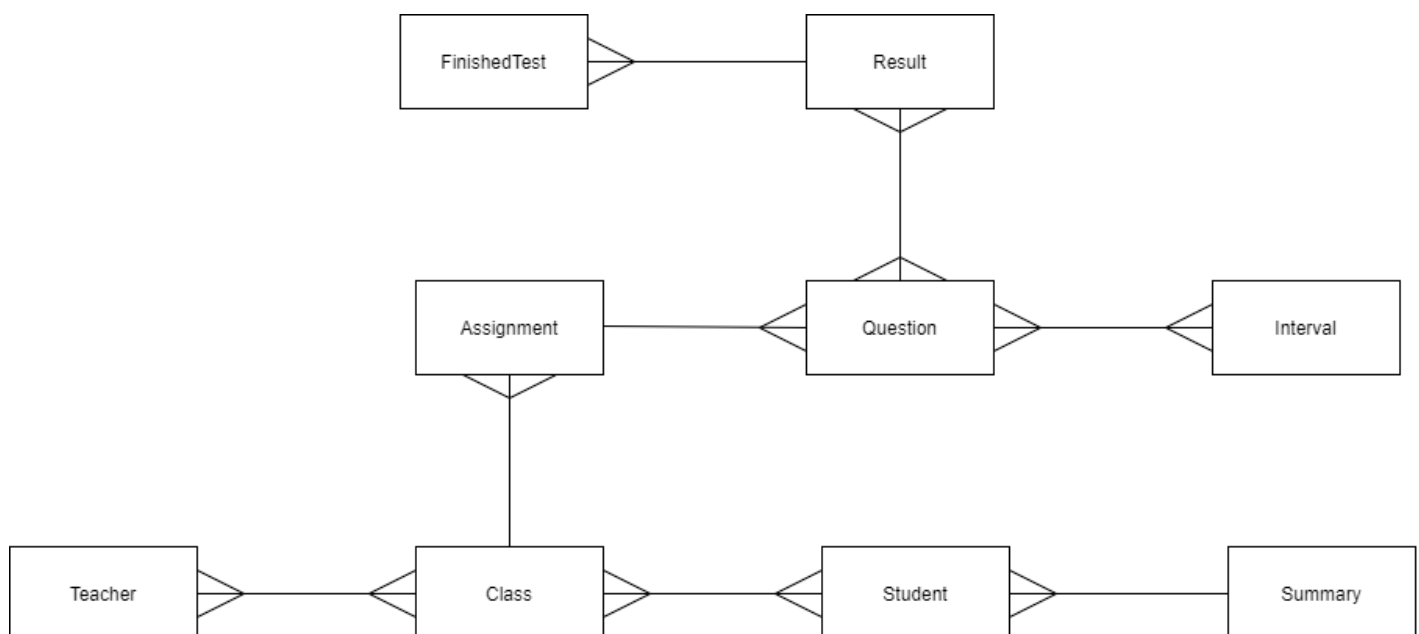
Result(TestID, StudentID, QuestionNumber, Correct)

FinishedTest(TestID, StudentID, PercentageScore, DateCompleted)

Summary(StudentID, TotalQuestions, TotalCorrect, AverageScore)

E-R Diagram

After normalising my database, I have now created an entity relationship diagram that shows the relationships between the tables I will create in the database.



Storage Specifications

The following specification sheets detail the data that will be stored in the new system. This will give me a better understanding of the data I will be handling in my system and how it will need to be handled. However, I doubt that the amount of data being processed in the system will be significant enough to consider special methods of data handling.

User Data

This sheet contains the volumetrics, regex, occurrence, and data sources for all of the data relating to the users in the system. This will be useful to consolidate how many users there will be in the system and how much data will be being stored for each user.

Volumetrics					
Document description		System	Document	Name	Sheet
User Data		Interval testing software	1	Katie Taylor	1
Stationery ref.		Size	Number of parts	Method of preparation	
User Data			1	Data entered by users and added to database	
Filing sequence		Medium		Prepared by	
Chronological		Database		Database	
Frequency of preparation		Retention period		Location of file	
Whenever a new user or class is created roughly once a year		Until the user/class is deleted		Database	
Volume	Minimum	Maximum	Av/Abs	Growth rate/fluctuations	
	5	15	10	Every year approx. 10 more users will be added as a new music class joins the college. Number of teachers is unlikely to change frequently.	
Users/receipts		Purpose			Frequency of use
System users		To store user data			Once per log in
Data Dictionary					
Ref	Name	Data Type	Regex	Occurrence	Source of data / description
1	StudentID	String	^[0-9]{6}\$	X20	CreateAccount
2	FirstName	String	[A-Za-z]+	X22	CreateAccount
3	Surname	String	[A-Za-z]+	X22	CreateAccount
4	Password	String	[A-Za-z]+	X22	CreateAccount
5	TeacherID	String	[A-Za-z]+	X2	System Input
6	ClassID	String	[A-Za-z]+	X2	Randomly Generated
7	ClassName	String	[A-Za-z]+	X2	Create Class

Interval Data

This sheet contains the volumetrics, regex, occurrence, and data sources for all the data relating to intervals. This will be useful to know how many intervals I will be storing information on and how many intervals I will be needing to fetch from the API in order to play out of the system.

Volumetrics					
Document description		System	Document	Name	Sheet
Intervals		Interval Testing Software	1	Katie Taylor	1
Stationery ref.		Size	Number of parts	Method of preparation	
Interval Data			1	Hard coded into database	
Filing sequence		Medium		Prepared by	
Ascending by semitones		Database		Katie Taylor	
Frequency of preparation		Retention period		Location of file	
once		Until software is deleted		Database	
Volume	Minimum	Maximum	Av/Abs	Growth rate/fluctuations	
	24	24	24	There is no fluctuation as no more intervals will be added over the time the system is running.	
Users/receipts		Purpose			Frequency of use
Students		To store the interval information			X10 per test
Data Dictionary					
Ref	Name	Data Type	Regex	Occurrence	Source of data / description
1	IntervalID	Integer	[0-9]+	X24	Interval List
2	Level	Integer	[1-4]+	X24	Interval List
3	Quality	String	[A-Za-z]+	X24	Interval List
4	Number	String	[0-9][A-Za-z]+	X24	Interval List
5	Semitones	Integer	[0-9]+	X24	Interval List

Test Data

This sheet contains the volumetrics, regex, occurrence, and data sources for all the data relating to the tests and the results of the tests in the system. This will give me an idea of how much data I will be handling per test per student.

Volumetrics					
Document description		System	Document	Name	Sheet
Test Data		Interval Testing Software	1	Katie Taylor	1
Stationery ref.		Size	Number of parts	Method of preparation	
Test Data			1	Put into database after teacher generates test and student takes test	
Filing sequence		Medium		Prepared by	
Chronological		Database		Student/Teacher	
Frequency of preparation		Retention period		Location of file	
Once per test generation and once per test taken.		Until student account deleted		Database	
Volume	Minimum	Maximum	Av/Abs	Growth rate/fluctuations	
	0	40	20	Students will probably take a regular of number tests each week not fluctuating.	
Users/receipts		Purpose			Frequency of use
Student/Teachers		To store the questions in tests and the data from the students taking the test			X1 per test
Data Dictionary					
Ref	Name	Data Type	Regex	Occurrence	Source of data / description
1	TestID	Integer	[0-9]+	X1	Randomly generated
2	QuestionNumber	Integer	[0-9]+	X10	Randomly generated
3	DueDate	Date	^(0?[1-9] [12][0-9] 3[01])[\\-](0?[1-9] 1[012])[\\-]d{4}\$	X1	Teacher input
4	Correct	Integer	[0-9]+	X10	Intervals
5	StartingNote	Integer	[0-9]+	X10	Randomly generated
6	Direction	String	[A-Za-z]+	X10	Randomly generated
7	TotalQuestions	Integer	[0-9]+	X1	Student
8	Total Correct	Integer	[0-9]+	X1	Student
9	AverageScore	Integer	[0-9]+	X1	Student
10	DateCompleted	Date	^(0?[1-9] [12][0-9] 3[01])[\\-](0?[1-9] 1[012])[\\-]d{4}\$	X1	Student

SQL DDL

In order to create the database, I have designed above, I will need to use DDL. I have written the DDL to create the database, tables, and attributes.

Creating the Database

```
"CREATE DATABASE IF NOT EXISTS `IntervalSoftwareDatabase`;"
```

Creating the Tables

```
"CREATE TABLE tblStudent
    (StudentID Char(6) NOT NULL PRIMARY KEY,
    FirstName Varchar(25),
    Surname Varchar(25),
    Password Varchar(25),
    ClassID Char(6))"
```

```
"CREATE TABLE tblTeacher
    (TeacherID Char(3) NOT NULL PRIMARY KEY,
    FirstName Varchar(25),
    Surname Varchar(25),
    Password Varchar(25))"
```

```
"CREATE TABLE tblResult
    (TestID Char(32) NOT NULL,
    StudentID Char(6) NOT NULL,
    QuestionNumber Integer NOT NULL,
    PRIMARY KEY(TestID, StudentID, QuestionNumber),
    Correct Boolean)"
```

```
"CREATE TABLE tblAssignment
    (TestID Char(32) NOT NULL PRIMARY KEY,
    TestName Varchar(20) NOT NULL,
    ClassID Char(6) NOT NULL,
    DueDate Date NOT NULL)"
```

```
"CREATE TABLE tblInterval
    (IntervalID Integer NOT NULL PRIMARY KEY,
    Level Integer,
    Quality Varchar(10),
    Number Varchar(4),
    Semitones Integer)"
```

```
"CREATE TABLE tblClass
    (ClassID Char(6) NOT NULL PRIMARY KEY,
    ClassName Varchar(15),
    TeacherID Char(3))"
```

```
"CREATE TABLE tblQuestion
    (TestID Char(32) NOT NULL,
    QuestionNumber Integer NOT NULL,
    PRIMARY KEY(TestID, QuestionNumber),
    IntervalID Integer,
    StartingNote Integer,
    Direction Varchar(4))"
```

```
"CREATE TABLE tblSummary
    (StudentID Char(6) NOT NULL PRIMARY KEY,
    TotalQuestions Integer,
    TotalCorrect Integer,
    AverageScore Integer)"
```

```
"CREATE TABLE tblFinishedTest
    (TestID Char(32) NOT NULL,
    StudentID Char(6) NOT NULL,
    PRIMARY KEY(TestID, StudentID),
    PercentageScore Integer NOT NULL,
    DateCompleted Date NOT NULL)"
```

Processes

Level Based Test Generation

```
Procedure GenerateIntervals()
  FOR n = 0 to 9
    IF Level = 1 THEN
      Random <- GenerateRandomNumber(0, 6)
      IF Random < 5 THEN
        Intervals(n) <- GenerateRandomNumber(0, 4)
      ELSEIF Random = 5 THEN
        Intervals(n) <- 6
      ELSEIF Random = 6 THEN
        Intervals(n) <- 11
      END IF
    END IF
    IF Level = 2 THEN
      DO
        randomNumber <- GenerateRandomNumber(0, 11)
      LOOP UNTIL randomNumber <> 5
      intervalIndex(n) <- randomNumber
    END IF
    IF Level = 3 THEN
      intervalIndex(n) <- GenerateRandomNumber(0, 11)
    END IF
    IF Level = 4 THEN
      intervalIndex(n) <- GenerateRandomNumber(0, 23)
    END IF
  END FOR
End Procedure
```

GenerateIntervals() is the procedure that generates the interval ID for the 10 intervals that will be tested in the test. It functions by generating a random number between two values dependent on the level 10 times. For level 1 the IDs 0-4, 6 or 11 can be generated as these are the easiest intervals. For level 2 any number between 0 and 11, excluding 5, can be generated as these are the non-compound major, minor or perfect intervals. Finally, if the level is 3 then it will generate numbers between 0 and 11 as these are all the non-compound intervals. For level 4 any number between 0 and 23 can be generated as there are 24 intervals in total and any interval can be generated.

Generating Starting Note and Direction

```

Procedure GenerateStartingNoteAndDirection()
  FOR n = 0 To 9
    SemitonesDB <- New DatabaseConnection("SELECT Semitones FROM tblInterval
    WHERE IntervalID = @IntervalID")
    Semitones(n) <- SemitonesDB.SelectData()
    IF Semitones(n) > 18 THEN
      IF GenerateRandomNumber(0, 100) mod 2 = 0 THEN
        StartingNote(n) <- GenerateRandomNumber(0, 11)
      ELSE
        StartingNote(n) <- GenerateRandomNumber(23, 35)
      END IF
    ELSE
      StartingNote(n) <- GenerateRandomNumber(0, 35)
    END IF
    IF StartingNote(n) + Semitones(n) > 35 THEN
      Direction(n) <- "DESC"
    ELSE IF StartingNote(n) - Semitones(n) < 0 THEN
      Direction(n) <- "ASC"
    ELSE
      IF GenerateRandomNumber(0, 100) mod 2 = 0 THEN
        Direction(n) <- "ASC"
      ELSE
        Direction(n) <- "DESC"
      END IF
    END IF
  END FOR
End Procedure

```

The procedure `GenerateStartingNoteAndDirection()` creates a starting note and an interval direction for each interval in the test that had been generated in `GenerateIntervals()`. This procedure will be able to be used for both level-based test generation and previous result test generation. It starts by connecting to the database with an SQL string that selects the number of semitones in the interval that had been selected in `GenerateIntervals()`. The program then sees if the number of semitones is greater than 18. This is because there are 36 pitches that will be able to be played out of the speaker and if the number of semitones is greater than 18 it is possible that the interval will not be able to be played if the starting note is randomised. Therefore, this if statement makes sure that the

starting note is always at the higher or lower end of the notes able to be played so that even if the number of semitones is the highest (24) the second note will be able to be played. If the number of semitones is less than 18 then the starting note is randomised. After this the program then generates the direction of the interval. First it checks if, when the semitones are added to the starting note, it will go out of the range of notes, if so then the direction will be descending. Then it checks if, when the semitones are taken away from the starting note, it will go out of the range of notes, if so then the direction will be ascending. If neither of these are the case, then the direction is randomised.

Previous Results Based Test Generation

Procedure GenerateIntervals()

```
FOR x = 0 to 23
```

```
    SQLString <- "SELECT COUNT(Correct) FROM tblResult  
                INNER JOIN tblQuestion  
                ON tblResult.TestID = tblQuestion.TestID  
                AND tblResult.QuestionNumber = tblQuestion.QuestionNumber  
                WHERE StudentID = @CurrentStudentID  
                AND IntervalID = @IntervalID  
                AND Correct = 1"
```

```
    CorrectCountDB <- New DatabaseConnection(SQLString)
```

```
    CorrectCount(x) <- CorrectCountDB.SelectData()
```

```
    SQLString <- "SELECT COUNT(Correct) FROM tblResult  
                INNER JOIN tblQuestion  
                ON tblResult.TestID = tblQuestion.TestID  
                AND tblResult.QuestionNumber = tblQuestion.QuestionNumber  
                WHERE StudentID = @CurrentStudentID  
                AND IntervalID = @IntervalID"
```

```
    TimesAttemptedDB <- New DatabaseConnection(SQLString)
```

```
    TimesAttempted(x) <- TimesAttemptedDB.SelectData()
```

```
    IF TimesAttempted(x) = 0 Then
```

```
        AverageSuccess(x) <- 0
```

```
    ELSE AverageSuccess(x) <- CorrectCount(x) / TimesAttempted(x)
```

```
    END IF
```

```
END FOR
```

(continued...)

```
AverageSuccess <- List(Of Decimals)
IF TimesAttempted(interval) = 0 THEN
    AverageSuccess.Add(0)
ELSE AverageSuccess.Add(CorrectCount(interval) / TimesAttempted(interval))
END IF

AverageSuccessWithIndex <- AverageSuccess.Select(Function(n, i) New With {.Num
= n, .Idx = i})

sortedSuccesses <- AverageSuccessWithIndex.OrderBy(Function(nwi) nwi.Num)
Intervals = sortedSuccesses.Take(10).Select(Function(x) x.Idx).ToArray()

FOR x As Integer = 0 To Intervals.Length - 1
    Randomize()
    randomIndex <- CInt(Int((Intervals.Length) * Rnd()))
    tempvalue1 <- Intervals(x)
    tempvalue2 <- Intervals(randomIndex)
    Intervals(x) <- tempvalue2
    Intervals(randomIndex) <- tempvalue1
END FOR

End Procedure
```

This procedure is another key algorithm that handle the generation of the 10 intervals for the test for the test based on previous results. The first part of the code, which is looped 24 times (1 for each interval), makes two calls to the database: one that counts the number of records where the student has got the interval correct and one where it counts the number of records where the student has been tested on the interval, regardless of the result. These are then assigned to their respective arrays with the interval as the index. Then, in order to make my algorithm as accurate as possible in seeing which intervals need to be tested a third array is created which normalises the number of times the interval has been guessed correctly by dividing it by the number of times it has been tested. If the number of times that it has been tested is 0 then the value for AverageSuccess is set to 0 (because you cannot divide by 0). The next part of the array then handles sorting the AverageSuccess array and then selecting the 10 intervals that have the smallest AverageSuccess. They are sorted so that the intervals will not be in ascending order. These are then assigned to intervals() and can be passed onto the GenerateStartingNoteAndDirection() procedure.

Validation

There will need to be some validation on the user inputs in the system in order to prevent errors. The system will not have that many user inputs as most selection will be done via drop downs however validation will be needed for when the users create accounts. Also, there will need to be presence checks for when a value from a dropdown must be selected.

Ref	Name	Validation Checks	Regex	Description	Error Message
1	Student Number	Type, Presence	^[0-9]{6}\$	Student number inputted when creating account	Please enter a valid 6-digit student number
2	Firstname	Type, Presence	[A-Za-z]+	First name inputted when creating account	Please enter a valid first name
3	Surname	Type, Presence	[A-Za-z]+	Surname inputted when creating account	Please enter a valid surname
4	Password	Presence	^[^s]*	Password inputted when creating account	Please enter a password
5	Confirm Password	Type, Presence	^[^s]*	Confirm password inputted when creating account	Please enter the password that matches the first password
6	Username	Presence	^[^s]*	Username inputted when logging in	Please enter a username
7	Password	Presence	^[^s]*	Password inputted when logging in	Please enter a password
8	Level	Presence	^[^s]*	Level dropdown when student creates level test/ teacher sets level test	Please select a level from the dropdown
9	Class Name	Presence	^[^s]*	Class name input when teacher creates class/When teacher sets test	Please enter a class name/Please select a class name from the dropdown
10	Due Date	Type, Presence	^d{1,2}\d{1,2}\d{4}\$	Due date selector for when teacher sets a class	Please select a due date/The due date cannot be before today
11	Quality/ Number	Presence	^[^s]*	Select boxes when student takes test	Please select a quality and number

SQL DML

Due to the fact that the data in the interval table will not be created or changed during the running of the program all of the values can be inserted into the table at once and do not need to be edited after this point. This SQL inserts the IntervalID, Quality, Number and Semitones into the table for all 24 intervals that will be tested.

```
"INSERT INTO tblInterval VALUES ('0', 'Minor', '2nd', '1'), ('1', 'Major', '2nd', '2'), ('2', 'Minor', '3rd', '3'), ('3', 'Major', '3rd', '4'), ('4', 'Perfect', '4th', '5'), ('5', 'Augmented', '4th', '6'), ('6', 'Perfect', '5th', '7'), ('7', 'Minor', '6th', '8'), ('8', 'Major', '6th', '9'), ('9', 'Minor', '7th', '10'), ('10', 'Major', '7th', '11'), ('11', '2', 'Perfect', '8th', '12'), ('12', '3', 'Minor', '9th', '13'), ('13', 'Major', '9th', '14'), ('14', 'Minor', '10th', '15'), ('15', 'Major', '10th', '16'), ('16', 'Perfect', '11th', '17'), ('17', 'Augmented', '11th', '18'), ('18', 'Perfect', '12th', '19'), ('19', 'Minor', '13th', '20'), ('20', 'Major', '13th', '21'), ('21', 'Minor', '14th', '22'), ('22', 'Major', '14th', '23'), ('23', 'Perfect', '15th', '24')"
```

Throughout the system data will be inserted into the tables. Below is the example of the SQL for adding the student data to the student table when a student creates an account which is representative of the SQL statements used to insert data into all the tables. As it will be user inputted data being used as variables the SQL statement has been parameterised. This has been done for all SQL statements that contain user inputted data in my program in order to program defensively and reduce the chance of SQL injection.

```
"INSERT INTO tblStudent VALUES (@StudentID, @FirstName, @Surname, @Password, @ClassID)"
```

Throughout the system I also will use SELECT statements in order to retrieve the data I need from the database. Below are some examples of SELECT statements I will use:

```
"SELECT COUNT(Correct) FROM tblResult
      INNER JOIN tblQuestion
      ON tblResult.TestID = tblQuestion.TestID
      AND tblResult.QuestionNumber = tblQuestion.QuestionNumber
      WHERE StudentID = @StudentID
      AND IntervalID = @IntervalID
      AND Correct = 1"
```



```
"SELECT ClassName, ClassID, COUNT(StudentID) As NumberOfStudents FROM tblclass  
LEFT JOIN tblStudent ON tblclass.ClassID = tblstudent.Class  
WHERE TeacherID = @ID  
GROUP BY ClassID"
```

```
"SELECT FirstName, Surname, tblStudent.StudentID, ClassName, AverageScore FROM  
tblStudent  
LEFT JOIN tblsummary ON tblstudent.StudentID = tblsummary.StudentID  
INNER JOIN tblClass ON tblstudent.Class = tblClass.ClassID  
WHERE TeacherID = @TeacherID"
```

```
"SELECT DISTINCT TestName, ClassName, DueDate FROM tblAssignment LEFT JOIN  
(SELECT TestID  
FROM tblResult  
WHERE StudentID = @StudentID)  
tblResult  
ON tblAssignment.TestID=tblresult.TestID  
INNER JOIN tblClass ON tblAssignment.ClassID = tblClass.ClassID  
WHERE tblResult.TestID is NULL AND tblAssignment.ClassID = @ClassID"
```

```
"SELECT Quality, Number, SUM(Correct = True OR Correct = False) As TotalTested,  
SUM(Correct = True) As TotalCorrect, CAST(SUM(Correct = True) / SUM(Correct = True OR  
Correct = False) * 100 As int) As PercentageCorrect  
FROM tblInterval  
INNER JOIN tblQuestion ON tblinterval.IntervalID = tblquestion.IntervalID  
INNER JOIN tblresult ON tblquestion.QuestionNumber = tblresult.QuestionNumber  
AND tblquestion.TestID = tblresult.TestID  
WHERE StudentID = @StudentID  
GROUP BY Quality, Number  
ORDER BY tblInterval.IntervalID ASC"
```

```
"SELECT AVG(PercentageScore) As PercentageScore, DateCompleted FROM tblfinishedtest
    WHERE StudentID = @StudentID
    GROUP BY CAST(DateCompleted AS DATE)
    ORDER BY DateCompleted ASC"
```

```
"SELECT Number FROM tblInterval
    INNER JOIN tblQuestion ON tblInterval.IntervalID = tblQuestion.IntervalID
    WHERE QuestionNumber = @QuestionCounter AND TestID = @ID"
```

```
"SELECT tblResult.QuestionNumber, Quality, Number, Correct FROM tblResult
    INNER JOIN tblQuestion ON tblResult.TestID = tblQuestion.TestID
    AND tblresult.QuestionNumber = tblquestion.QuestionNumber
    INNER JOIN tblInterval ON tblquestion.IntervalID = tblinterval.IntervalID
    WHERE tblResult.TestID = @TestID AND StudentID = @StudentID"
```

All of the statements shown above will be used to fetch data directly to the program in order to be used in calculations or will be used to display the data to DataGrids or charts to display the data from the database to users.

I also will use some UPDATE statements. These are used to update information already in a record that is already in the database. For this example, it is changing the class of the student as students may leave a join classes.

Below are some examples of these:

```
"UPDATE tblStudent SET Class = null WHERE StudentID = @ID"
```

```
"UPDATE tblStudent SET Class = @ClassID WHERE StudentID = @StudentID"
```

Communication

Database Connection

In order to connect to and use the database the software must connect to it. This is done through a connection string that will have the user, the name of the database, the port, and the password. For the purposes of testing the connection string has been hard coded – however in the implementation of the code the connection string will be saved to and read from a file. The database connections have also all been programmed defensively with try catch statements to reduce crashes. Below is the code for the connection string and for opening a connection to the database:

```
Imports MySql.Data.MySqlClient

Dim MyCommand As MySqlCommand
Dim MyConnection As MySqlConnection
Dim MyReader As MySqlDataReader
Dim DDLstr As String
Dim SQLstr As String
Dim ConnStr As String =
"server=localhost;user=testuser;Database=neatesting;port=3306;password=testing123;"

Sub New(ByVal selectString As String)
    SQLstr = selectString
    OpenDatabase()
End Sub

Public Sub OpenDatabase()
    Try
        MyConnection = New MySqlConnection(ConnStr)
        MyConnection.Open()
    Catch
        MsgBox("Could not connect to the specified database.")
    End Try
End Sub
```

The SQL statements listed previously must also be used when connecting to the database to execute the SQL statement. Below is an example of how the SQL statement saving to a user table would be processed:

```
Try
    MyCommand = New MySqlCommand(SQLstr, MyConnection)
    MyCommand.Parameters.AddWithValue("@ID", ID)
    MyCommand.Parameters.AddWithValue("@FirstName", FirstName)
    MyCommand.Parameters.AddWithValue("@Surname", Surname)
    MyCommand.Parameters.AddWithValue("@Password", Password)
    MyCommand.ExecuteNonQuery()
Catch ex As Exception
    MsgBox(ex.Message)
End Try
MyConnection.Close()
```

Testing Strategy

In order to begin implementing my design I also need to have worked out a testing strategy so I have a clearer idea of what I will be testing so that I can make sure that the code will be able to pass those tests as expected.

Firstly, I have detailed the testing data that I will be using so that I have a clear idea of what data I will be using for tests and therefore an understanding of what should be outputted for each test.

I have also outlined which processes I will be testing and how I will be testing them (white box, black box) and how I will be testing the GUI and IO (using TEX inputs).

I will also describe how I will carry out the Beta testing of the system. This will include who will be testing my system, with what data they will be testing the system and a list of feedback questions to ask them about the performance of the system.

Processing Test Data

Name	Value	Tests Used In
Student ID	192267	1, 3
(Student) First Name	Paul	1
(Student) Surname	Rudd	1
(Student) Password	HelloWorld	1, 3
Teacher ID	ABC	2, 4
(Teacher) First Name	Abby	2
(Teacher) Surname	Cole	2
(Teacher) Password	Computer	2, 4
Class ID	213213	7
Assignment Name	Test 1	9
Level	Level 1	10
Class Name	Abby's Class 1	14, 16, 17
Intervals	Major 2 nd , Minor 3 rd , Perfect 4 th , Perfect 5 th , Minor 7 th , Perfect 8 th , Major 10 th , Perfect 11 th , Major 14 th , Perfect 15 th	16
Date Due	24/03/21	16, 17
Level	Level 3	17
Manual Assignment Name	Manual Test	16
Auto Assignment Name	Auto Test	17
Student Graph	Katie Taylor	19

Process Testing Plan

No.	Name	Description	Test Type
1	Create student account	Input data into create student account and show data being saved into database.	Black box
2	Create teacher account	Input data into create teacher account and show data being saved into database.	Black box
3	Log in to student account	Input username and password and show that it logs into correct account.	Black box
4	Log in to teacher account	Input username and password and show that it logs into correct account.	Black box
5	Log out of student account	Show that it goes back to main menu and you can log in as a new user.	Black box
6	Log out of teacher account	Show that it goes back to main menu and you can log in as a new user.	Black box
7	Student join class	Enter class ID and show that class ID has been added to the student's database record.	Black box
8	Student leave class	Press button and show that class has been removed from student's database record.	Black box
9	Student take assignment	Click on assignment and show how intervals being played match those in the database. Step through interval generation.	White box
10	Student take manual test	Click on level and step through generation of intervals and generation of starting notes/direction.	White box
11	Student take automatic test	Calculate which intervals should be tested then step through and show these are the ones being selected.	White box
12	Student view results graph	Calculate data points that should be on the graphs from database then step through generation.	White box
13	Student view interval data	Calculate percentages and numbers of intervals from database and step through generation.	White box
14	Teacher create new class	Show that new class is now in database.	Black box
15	Teacher view class list	Calculate number of students in each class and show it matches.	Black box
16	Teacher set manual test	Select intervals and step through them being assigned as the intervals and saved to database.	White box
17	Teacher set automatic test	Click on level and step through generation of intervals and generation of starting notes/direction. Show that test shows up as assignment and in database.	White box
18	Teacher view student results	Show that percentages and names match percentages in the database.	Black box
19	Teacher view student graph	Calculate the data points for the student graph and see if they match the graph output.	Black box

GUI/IO Test Data

Name	Value	Tests Used In
Valid Student ID	196666	1, 4
Valid Student ID	193217	3, 5
(Student) First Name	Joel	1, 2, 3
(Student) Surname	Peters	1, 2, 3
(Student) Password	NEA123	1, 2, 4
Extreme (Student) Password	K	3
Incorrect Password	HelloWorld	5
Invalid Student ID	B89	2
Valid Teacher ID	ABC	1, 4
Valid Teacher ID	ABC	3, 5
(Teacher) First Name	Alice	1, 2, 3
(Teacher) Surname	Cone	1, 2, 3
(Teacher) Password	TeacherPass	1, 2, 4
Extreme (Teacher) Password	2	3
Invalid Teacher ID	AAA	2
Valid Level	Level 1	6
Invalid Level	(no level selected)	7
User for auto test (with data)	192344	8
User for auto test (no data)	196666	9
Test boxes	Quality = Major, Number = 2nd	10
Class ID	213213	11
Class ID (invalid)	hello	12
10 intervals selected	Major 2 nd , Minor 3 rd , Perfect 4 th , Perfect 5 th , Minor 7 th , Perfect 8 th , Major 10 th , Perfect 11 th , Major 14 th , Perfect 15 th	13
5 intervals selected	Major 2 nd , Minor 3 rd , Perfect 4 th , Perfect 5 th , Minor 7 th	14
Class name	My Class	13, 14, 16
Due Date	24/03/21	13, 14, 18
Class name (valid)	Alice's L6	15
Class name (extreme)	U	17
Due Date (extreme)	07/03/21	19

GUI/IO Testing Plan

No.	Description	Data Type	Expected Result
1	Create Account	Typical e.g. correct data types for all inputs	Message will state that account has been created and they will be taken back to launch page
2	Create Account	Erroneous e.g. invalid username	Error message will occur, and user will be prompted to re-enter data
3	Create Account	Extreme e.g. 1 letter password	Message will state that account has been created and they will be taken back to launch page
4	Log in	Typical e.g. correct username and password	Account will log in and user will be taken to account tab of main menu
5	Log in	Erroneous e.g. incorrect username and password	Message prompting to enter valid details.
6	Generate Level Test	Typical e.g. Level 1	Test is generated and started
7	Generate Level Tests	Erroneous e.g. No level selected	Error message prompting user to select a difficulty level
8	Student Generate Automatic Test	Typical e.g. have answered many tests previously	Will create test out of the intervals the student has guessed incorrectly the most
9	Student Generate Automatic Test	Extreme e.g. has not answered any questions before	Will create a test with the first 10 intervals on the interval list
10	Student Takes Test	Typical e.g. checks a box for both a quality and number	The answer will be marked as correct or incorrect and then it will move to next question.
11	Student Takes Test	Erroneous e.g. only quality box checked	Message will prompt to select both a quality and number and it will stay on the same question.
12	Student Joins Class	Typical e.g. valid class ID	Screen will change to a list of assignments for that class
13	Student Joins Class	Erroneous e.g. invalid class ID	Message will prompt student to enter a valid Class ID
14	Manual Teacher Test	Typical e.g. 10 intervals selected	The test will be set to the class
15	Manual Teacher Test	Erroneous e.g. less than 10 selected	Message will ask for 10 intervals to be selected before test can be set
16	Create new class	Typical e.g. type in a class name	A message will appear confirming the class has been created
17	Create new class	Erroneous e.g. class name already used	Message will ask for teacher to enter a new class name
18	Create new class	Extreme e.g. one letter class name	A message will appear confirming the class has been created
19	Set due date for test	Typical e.g. a week from the current date	Test will be created and set
20	Set due date for test	Extreme e.g. the current day	The test will be created and set
21	Navigation	Typical e.g. buttons work	Throughout testing when buttons or tabs are clicked it correctly navigates to desired location

Beta Testing

Who?

My software will be beta tested by Paul Clifford who is the A Level music teacher and my third party. He will test both the teacher and the student functionalities.

Data

He will be using data of his own decision in order to allow for him to enter types of extreme or erroneous data that I may not have considered and therefore have the capability to crash the program.

Feedback Questions

For the student account:

1. Were you able to create a student account?
2. Were you able to log into the student account you created?
3. Were you able to view the account information?
4. Were you able to create and take a level-based test?
5. Were you able to create and take a previous result-based test?
6. Were the results of these tests then able to be viewed in the results page?
7. Were you able to take a teacher assignment?
8. Was the teacher assignment then stored in the 'previous assignments' section?

For the teacher account:

9. Were you able to log in?
10. Were you able to view the account information?
11. Were you able to create a new class?
12. Were you able to set a level-based test to a class?
13. Were you able to set a manual test to a class?
14. Were you able to view the student summaries and graphs?

Overall:

15. Did you encounter any errors or crashes during your use of the program: if so were where they and what were they caused by?
16. Are there any elements of the program that were confusing and needed better labelling or explaining?
17. Are there any other features that you feel should be added to the software that are vital for it to be functioning?
18. Any other thoughts on the program?

Testing

For the testing of my completed system I have completed the tests as outlined in my testing strategy. I have recorded myself performing these tests and uploaded the video and have given the timestamps for each test in the video.

The tests I have performed serve as to prove that my system works as intended and can perform all of the functions I outlined in my requirements. I have done this by testing inputs/outputs as well as black box and white box testing processes in order to show my algorithms functioning.

GUI/IO Testing

The full video of the GUI/IO testing can be found at:

<https://www.youtube.com/watch?v=pdmUeaO1Km0>

No.	Purpose	Type	Test Data	Timestamp
1	Create Account: when all data is entered correctly an account is created	Typical	StudentID: 196666, First Name: Joel, Surname: Peters, Password: NEA123, TeacherID: ABC, First Name: Alice, Surname: Cone, Password: TeacherPass	0:09
2	Create Account: if incorrect data is entered it is validated and an account is not created	Erroneous	StudentID: b89, First Name: Joel, Surname: Peters, Password: NEA123	1:20
3	Create Account: if extreme data is entered an account will still be created	Extreme	StudentID: 193217, First Name: Joel, Surname: Peters, Password: K	1:56
4	Log In: if correct data is used then the user will be able to log in	Typical	Username: ABC, Password: TeacherPass	2:23
5	Log In: if an invalid username is entered then it will not login	Erroneous	Username: 196667, Password: NEA123	2:44
6	Level Test: if a level is selected then a test of that level is created and run	Typical	Level: Level 1	3:12
7	Level Test: if a level is not selected then a test is not created or run	Erroneous	Level: null	3:33
8	Auto Test: if a student who has many previous tests uses an auto test it will run	Typical	StudentID: 192344	3:52

9	Auto Test: if a student who has not taken a test before uses an auto test it will run	Extreme	StudentID: 196666	4:15
10	Take Test: if a quality and number are selected then it will mark and move to next question	Typical	Quality: Perfect, Number: 8th	4:43
11	Take Test: if only one box is checked then it will not mark and request a full answer	Erroneous	Quality: Perfect, Number: null	5:15
12	Join Class: if correct class ID is entered student will join that class	Typical	Class ID: 213213	5:39
13	Join Class: if invalid class ID is entered then no class will be joined	Erroneous	Class ID: hello	6:09
14	Teacher manual test: if 10 intervals are selected then test is created and set	Typical	Intervals: Major 2 nd , Minor 3 rd , Perfect 4 th , Perfect 5 th , Minor 7 th , Perfect 8 th , Major 10 th , Perfect 11 th , Major 14 th , Perfect 15 th , Class: My Class, Due Date: 24/03/2021	6:32
15	Teacher manual test: if 10 intervals are not selected then test is not created	Erroneous	Intervals: Major 2 nd , Minor 3 rd , Perfect 4 th , Perfect 5 th , Minor 7 th , Class: My Class, Due Date: 24/03/2021	7:40
16	New Class: if a valid class name then class created	Typical	Class Name: Alice's L6	8:12
17	New Class: If a class name that is already used is entered then a new class name is required	Erroneous	Class Name: My Class	8:37
18	New Class: if a one letter class name is entered it is accepted by the system	Extreme	Class Name: U	8:59
19	Due Date: if a date in the future is set then the test is created with that due date	Typical	Level: Level 2, Class: Alice's L6, DueDate: 24/03/2021	9:16
20	Due Date: if the current date is selected then it is still used as the due date	Extreme	Level: Level 2, Class: Alice's L6, DueDate: 07/03/2021	9:40
21	Navigation: all the buttons and tabs, when clicked, navigate to the correct location, and are clearly labelled	Typical	n/a	10:02

Process Testing

The full video of the process testing can be found at:

<https://www.youtube.com/watch?v=D2QXt1MbU1g>

No.	Purpose	Test Data	Expected Results	Actual Results	Timestamp
1	Create student account	StudentID: 192267, Name: Paul Rudd, Password: HelloWorld	Test data is saved to the database	Test data is saved to the databased	0:06
2	Create teacher account	TeacherID: ABC, Name: Abby Cole, Password: Computer	Test data is saved to the database	Test data is saved to the database	0:52
3	Log in to student account	Username: 192267, Password: HelloWorld	Logs into correct account	Logs into correct account	1:31
4	Log in to teacher account	Username: ABC, Password: Computer	Logs into correct account	Logs into correct account	2:06
5	Log out of student account	n/a	Directs user back to launch page	Directs user back to launch page	2:48
6	Log out of teacher account	n/a	Directs user back to launch page	Directs user back to launch page	3:06
7	Teacher create new class	Class Name: Class 1	New class is created with random 6-digit code and is saved to databased	New class is created with code 584302 and is saved to database	3:23
8	Student join class	Class ID: 584302	Assignment tab screen changes to tables and class ID is added into student record	Assignment tab screen changed to tables and 584302 was added into student record	3:57
9	Teacher view class list	n/a	Table shows class name, class ID and the number of students in the class from database.	Table shows Class 1, 584302, and 1 which match data in database	4:29
10	Teacher set auto test	Level: Level 3	An array on 10 intervals with interval IDs	The intervals (5, 3, 0, 5, 2, 0, 3, 6, 2, 2)	5:03

		Class: Class 1 Due Date: 24/03/2021 Test Name: Test 1	between 0 and 11 and random starting notes and directions are generated then saved to the question table and assignment table in database.	were generated which are all between 0 and 11. The starting notes and directions were all generated and then assigned to the database in the question and assignment tables.	
11	Teacher set manual test	Intervals: Minor 2 nd , Major 2 nd , Major 3 rd , Perfect 4 th , Perfect 5 th , Major 6 th , Perfect 8 th , Major 9 th , Major 10 th , Perfect 11 th , Minor 13 th , Class: Class 1 Due Date: 24/03/2021 Test Name: Test 2	The array of interval IDs match those of the inputted intervals. Random starting notes and directions are calculated then saved to the question and assignment tables in the database.	The interval array was set to (0, 1, 3, 4, 6, 8, 13, 15, 16, 19) which match the intervals selected. Starting notes and directions were generated and then saved to the question and assignment tables in the database.	11:28
12	Student take assignment	Assignment: Test 1	The information for the test selected is taken from the database. Then the program will open the take test form and the student will be able to listen to the intervals and guess them. It then saves results to the database.	Takes the intervals (5, 3, 0, 5, 2, 0, 3, 6, 2, 2) from the database and then opens the test and plays each interval for each question. The results are then displayed in a table and saved to the database.	15:05
13	Student take manual test	Level: Level 2	A test of level 2 is generated: the intervals will be between 0 and 11. It will then generate a starting note a direction for each interval, save to database, then run the take test class and allow the student to take the test and saves score to database.	A test of level 2 is generated. It generated a starting note a direction for each interval, save to database, then run the take test class and allowed the student to take the test and saved score to database.	25:16

14	Student take auto test	StudentID: 192344	Generates an array of intervals by finding the 10 lowest scoring intervals for that student in the database then randomly sorting the intervals so they are not in order. Then calculates starting note and direction, saves to database, and runs test as usual and results are saved to database.	It got the average score for each interval from the database and assigned it to an array. It then selected the 10 lowest scoring of these intervals (5, 7, 9, 12, 13, 14, 15, 16, 17, 18) and then sorted them so they were in the order (12, 16, 15, 14, 7, 9, 18, 5, 17, 13). It then calculated the starting note and direction and saved the questions to the database then ran the test.	26:57
15	Student view graphs	StudentID: 192344	Produces a graph with all dates and scores plotted then when the button is pressed it groups the data for all the tests on each day and plots the average score for that day.	The graph is shown for all the student's data points and then when the button is clicked it showed the grouped average data points.	31:54
16	Student view interval data	StudentID: 192344	Gets information for each interval and how well it has been answered by the student from the database.	Gets information for each interval and how well it has been answered by the student from the database.	33:22
17	Teacher view student results	n/a	Gets grid of all the students and their average score and class for all students that are in a class owned by the teacher that is logged in.	Gets grid of all the students and their average score and class for all students that are in a class owned by the teacher that is logged in.	34:48
18	Teacher view student graph	StudentID: 192344	Creates the grouped graph for the student selected.	Creates the grouped graph for the student selected.	35:50
19	Student leave class	StudentID: 192267	Class is removed from student record	Class is removed from student record	36:40

Beta Testing

The beta testing was carried out by **Paul Clifford**, the project's third party, on the 10th of March 2021. As he will be the primary user of the software it is important that he is able to test it and give feedback where necessary.

For the student account:

1. Were you able to create a student account? **Yes, I was**
2. Were you able to log into the student account you created? **Yes, I was**
3. Were you able to view the account information? **Yes, I was**
4. Were you able to create and take a level-based test? **Yes, I was**
5. Were you able to create and take a previous result-based test? **Yes, I was**
6. Were the results of these tests then able to be viewed in the results page? **Yes, they were.**
7. Were you able to take a teacher assignment? **Yes, I was**
8. Was the teacher assignment then stored in the 'previous assignments' section? **Yes, it was.**

For the teacher account:

9. Were you able to log in? **Yes, once I had added my teacher ID to the list of valid teacher IDs.**
10. Were you able to view the account information? **Yes, I was**
11. Were you able to create a new class? **Yes, I was**
12. Were you able to set a level-based test to a class? **Yes, I was**
13. Were you able to set a manual test to a class? **Yes, I was**
14. Were you able to view the student summaries and graphs? **Yes, I was**

Overall:

15. Did you encounter any errors or crashes during your use of the program: if so where were they and what were they caused by?

I did not encounter any errors or crashes during my time testing out the software.

16. Are there any elements of the program that were confusing and needed better labelling or explaining?

Overall, the user interface was easy to understand and navigate without needing it to be explained to me. The design of the user interface could have been cleaner in order to make it even easier to navigate and perhaps the window could be bigger so that the screen did not feel as cramped.

17. Are there any other features that you feel should be added to the software?

All of the features that are currently there make for a good and useful system that achieve the software that I required. Perhaps even more features could be added such as a test scheduling system and a chord recognition mode to make it even more useful, but they are not vital for running and the current features are more than sufficient.

18. Any other thoughts on the program?

I really like the feature that allows students to view their progress on each individual interval as this was not something I asked for, but I think will be really useful for the students.

Evaluation

In my evaluation I will detail how I would have changed my current solution if I were to do it again, any additional features that I could add which are outside of the requirements and a detailed breakdown of each requirement and how well I achieved each one. This will be to assess to what extent I have fulfilled the requirements of my project and to what extent I have successfully created my project.

Revisiting the Problem

If I were to do this project again, however, I would consider making the program web-based or changing the software used so that it could be accessed more easily by the students and could have a more modern and cleaner user interface.

However, if I were going to keep the program in visual studio if I were to do the project again, I would choose to enable a full screen mode so that the user interface could feel less tight for space. I would also choose a sans serif font for the text in the program to make it look cleaner and take account for the fact that the text becomes slightly pixelated when the program runs.

I would also consider finding another method of outputting the intervals as, although the current method of using the console sounds works and fits the requirements, the sound sometimes sounds poorer quality in the higher and lower registers. Another way to work around this problem would be to decrease the range of notes that the system plays but keep the current system sound output.

Additional Features

Additional features that I could add onto my project that I have considered include a test scheduling system. This would allow teachers to set assignments for the students in advance and the assignments are only shown to the students at a certain time that is specified by the teacher. This could be useful for setting a weekly homework or tests and could extend the functionality of the program and make it even more useful for the third party.

Another new feature could be different testing modes and expanding the software beyond just interval recognition. This could include chord recognition modes in which multiple notes could be played at once and the student would need to identify the name of the chord and in harder versions identify which inversion of the chord it is.

The program could also be made more interactive between the students not just between the student and the teacher. There could be a student leader board with the students being ranked on their weekly or daily scores and how many tests they have taken. This would increase student motivation to use the program and make it more fun to use.

I could also introduce badges/achievements for the students. These achievement could be earned with use of the program (e.g. take 10 level 1 tests, earn 100% in a test) and would increase the student's desire to use the program as it would make it more fun and competitive.

Achievement of Requirements

This table lists the requirements and how well I believe I have achieved each requirement. Refer to [pages 29-30](#) for the original detailed requirements list and [pages 64-68](#) for the testing.

KEY

COLOUR	MEANING
	Requirement Completed
	Requirement Partially Completed
	Requirement Not Completed

No.	Details	Achievement	Test Reference	Justification
1.1	Multiple clients should be able to connect		All process	As seen throughout the videos I am using MySQL which allows a connection from any client
1.2	The clients should be able to connect simultaneously		All process	MySQL can handle multiple clients connecting at once and for the number of users of this program, handling simultaneous connection is not an issue
2.1	Navigation tools must be labelled descriptively		GUI/IO 21	As seen throughout the videos, all the buttons and navigation tools are descriptively labelled or use a clear symbol to describe what they do
2.2	Program interface must be laid out simply and clearly		All GUI/IO	All the text is legible, and the fonts are consistent however the screen can at times look crowded and slightly messy
2.3	There must be a simple and cohesive colour scheme		All GUI/IO	I used a grey and blue theme throughout, so the colours are consistent but not overwhelming
3.1	When a student creates an account the relevant data is stored		GUI/IO 1	The create account screen captures in the student ID, name and password for the new account and saves it to variables
3.2	When a teacher creates an account the relevant data is stored		GUI/IO 1	The create account screen captures in the teacher ID, name and password for the new account and saves it to variables
3.3	The create account data must be saved to the database		Process 1/2	All the data captured are saved to the relevant fields in the teacher or student tables in the database
4.1	To log in a username and password must be typed into textboxes		GUI/IO 4	The log in menu has two textboxes: one for the username and one for the password. The user can enter their details into them to log in

4.2	The log in data entered should be checked against the database		Process 3/4	The program uses an SQL statement to see if the username and password both match an entry in the database
4.3	If the username and password match an account then they can log in		Process 3/4	If the username and password matched the main menu of the program is opened logged in with the provided username
5.1	When creating a class the teacher is prompted to enter a class name		GUI/IO 16	In the create new class form there is a textbox to enter a class name and a class name must be entered to create the class
5.2	A random class code is generated which can be given to students		Process 7	A random code is generated once the button is clicked and can be seen in the table on the class tab
5.3	The class code is stored in the database and linked with the teacher		Process 7	The class code is stored in the class table of the database in a record with the class name and the teacher ID associated with it
6.1	The teacher can select either a randomly generated test or a manual test		GUI/IO 14	There are two tabs for setting assignments: manual and automatic. They will each set the different type of test
6.1.1	For the randomly generated test the teacher chooses a difficulty level		Process 10	There is a combo box for the level on the automatic test and the teacher can select the level from a dropdown
6.1.2	An algorithm determines the 10 intervals to be tested based on the level		Process 10	It generates the intervals using the Level Test class and assigns them to a list and saves them to the database
6.1.3	For the manual test, the teacher can select 10 intervals to be tested		Process 11	There is a list box on the manual test tab which enables the teacher to tick the 10 intervals they want to be tested
6.2	The teacher can select what class to set the test to		GUI/IO 14	There is a combo box which contains a list of all the teacher's classes and the teacher can select which class to assign the test to from the drop down
6.3	The teacher can select a due date for the test		GUI/IO 14	There is a date picker which allows the teacher to select a date from the current date and beyond
7.1	If an automatic test is taken by a student an algorithm will determine what interval is played based on difficulty level		Process 13	The student can select the level of the test from the dropdown and generate the test. The intervals are determined by selecting a random interval from within the intervals of that difficulty level

7.2	A random number index will be taken from an array of notes and be selected as the starting note		Process 13	The starting note is selected by assuring that the interval can start on the note selected and then choosing a random number out of the 36 notes available
7.3	A direction for the interval will be selected		Process 13	The system checks that if the interval were to be ascending/descending it would stay in the note range and then selects the direction accordingly. If it could go either way then a random direction is chosen
8.1	The two notes that are required to play the interval that has been generated are taken from the database		Process 12/13/14	The starting note and the starting note plus or minus the number of semitones in the interval (depending on the direction) is passed to the Interval class to be played
8.2	The interval is played using the computer's speaker using console sound system		Process 12/13/14	The console function is used to play the notes of the specified frequency that is associated with the two notes passed in
9.1	The test will run – play the interval and be able to select answer from list		Process 12/13/14	There is a button to play the interval up to two times and two list boxes which contain all the options for qualities and numbers of the intervals
9.2	The data from the test is saved to the database		Process 12/13/14	Once all 10 questions have been answered the achievement on the questions is saved to the Results table, the test is saved to the Finished Test table and the values in the Summary table are updated
9.3	The student will be able to view an overview page and are given a percentage grade		Process 12/13/14	After the test has run a table with the results for each question is shown and then a percentage for the test is in a label at the top of the form
10.1	Students can select an assignment to take from a list of assignments		Process 12	In the Assignments tab there is a table under current assignments that details all of the assignments they currently need to take which can be clicked on to take
10.2	The assignment will run as in requirement 9		Process 12	When the assignment is clicked on the test will open and run
11.1	The student can take a randomly generated test by selecting a level		Process 13	In the manual test tab the students can select a test level from the drop down and click run test

11.2	The test will run as in requirement 9		Process 13	When the button is clicked the test will open and run
12.1	Student can take a test based on previous results where computer tests 10 most misidentified intervals		Process 14	The algorithm takes the results for each interval and puts the 10 worst intervals in an array which is then saved to the database as the list of intervals for the test
12.2	The test will run as in requirement 9		Process 14	Once the intervals have been generated then the test will open and run
13.1	A graph is created with from all previous test results		Process 15	The graph plots all of the percentages of each test against the date they were taken to show progression over time
13.2	A button can be pressed to change the graph to be grouped by date		Process 15	When the button is pressed the graph changes to show the average score for each day plotted instead of each individual test
13.3	The graph can be viewed by students by clicking on their results page		Process 15	The graph is displayed on the My Results tab on the student menu
13.4	The graph can be viewed by teachers by clicking on the student pages and selecting the student		Process 17/18	On the teacher's Student Results tab is a list of all of the students that are a member of any of their classes. If they click on the student then the graph is displayed

Conclusion

As shown in the table above, I have completed all of my requirements for my project. This shows that I have successfully implemented my design and have created a working system that has fully achieved all of the wishes of my third party as established in my interview with him during the analysis stage of the process.

The feedback I got from my third party when he was carrying out the beta testing also serves to prove that the system fulfils the expectations of the third party, as he was happy with all elements of the functionality of the program and only had some criticisms for the user interface.

When considering the elements of the system I would redo if I were to revisit the problem, there were no elements that were vital to the functioning of the software that I felt as though I needed to redesign, and it was mainly aesthetic changes.

Overall, I am very happy with how my program has turned out and I believe it suitably fulfils the brief given by the third party and matches the design and the requirements I created before I began to implement the solution.

Bibliography

Anon., n.d. *Chart Axes*. [Online]

Available at: <https://help.syncfusion.com/windowsforms/chart/chart-axes>

[Accessed 22 January 2021].

Anon., n.d. *Combobox Text and Value*. [Online]

Available at: <http://net-informations.com/q/faq/combovalue.html>

[Accessed 18 January 2021].

Anon., n.d. *Console.Beep Method*. [Online]

Available at: <https://docs.microsoft.com/en-us/dotnet/api/system.console.beep?view=net-5.0>

[Accessed 27 December 2020].

Anon., n.d. *DateTimePicker Minimum Date*. [Online]

Available at: <https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.datetimepicker.mindate?view=net-5.0>

[Accessed 29 January 2021].

Anon., n.d. *Generate Unique ID*. [Online]

Available at: <https://stackoverflow.com/questions/21433670/auto-generate-unique-id-within-the-constructor>

[Accessed 30 December 2020].

Anon., n.d. *Get DataGridView Cell Value*. [Online]

Available at: <https://stackoverflow.com/questions/2447358/return-get-datagridview-cell-value-in-vb-net>

[Accessed 28 January 2021].

Anon., n.d. *How To Populate Chart with SQL*. [Online]

Available at: <https://www.youtube.com/watch?v=h3nHrZxDub8>

[Accessed 18 January 2021].

Anon., n.d. *Insert Data into DataGridView*. [Online]

Available at: <https://stackoverflow.com/questions/16530686/vb-net-insert-datagridview-contents-into-database>

[Accessed 5 January 2021].

Anon., n.d. *Interval (music) Wikipedia*. [Online]

Available at: [https://en.wikipedia.org/wiki/Interval_\(music\)](https://en.wikipedia.org/wiki/Interval_(music))

[Accessed 15 September 2020].

Anon., n.d. *Parameterized Queries*. [Online]

Available at: <https://www.codeguru.com/columns/vb/using-parameterized-queries-and-reports-in-vb-net-database-applications.htm>

[Accessed 25 November 2020].

Anon., n.d. *Populate DataTable with SQL*. [Online]

Available at: <https://www.aspsnippets.com/Articles/Fill-Populate-DataTable-using-SqlDataAdapter-in-C-and-VBNet.aspx>

[Accessed 21 January 2021].

Anon., n.d. *Set Control Property While Looping Through Controls*. [Online]
Available at: <https://stackoverflow.com/questions/56757090/set-control-property-while-looping-through-controls>
[Accessed 4 January 2021].

Anon., n.d. *SQL Alias*. [Online]
Available at: https://www.w3schools.com/sql/sql_alias.asp
[Accessed 8 January 2021].

Anon., n.d. *SQL Count, Avg, Sum*. [Online]
Available at: https://www.w3schools.com/sql/sql_count_avg_sum.asp
[Accessed 8 January 2021].

Anon., n.d. *SQL Distinct*. [Online]
Available at: https://www.w3schools.com/sql/sql_distinct.asp
[Accessed 8 January 2021].

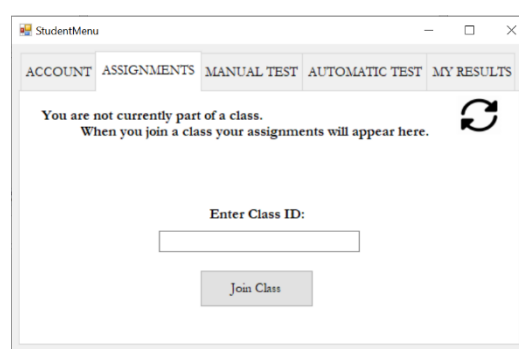
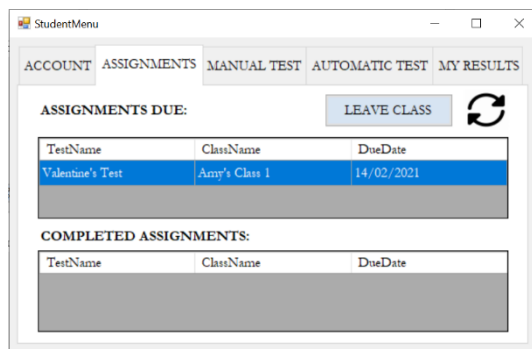
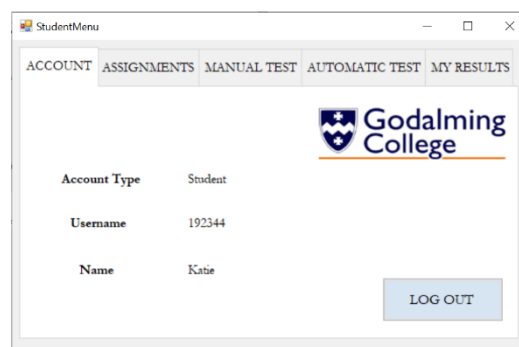
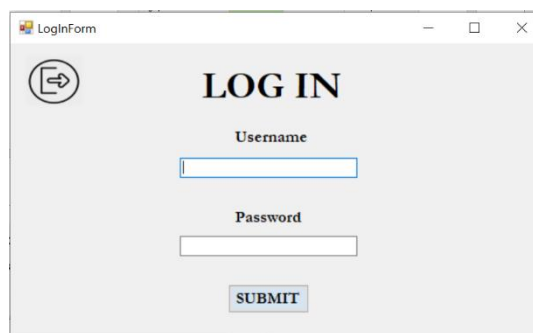
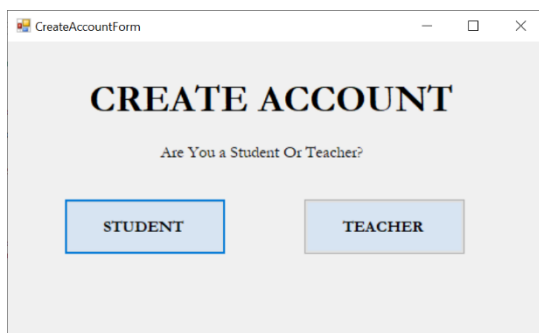
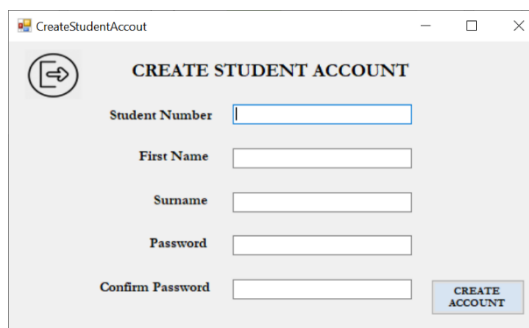
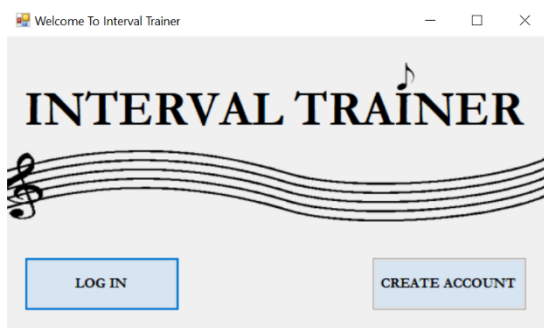
Anon., n.d. *SQL Subquery*. [Online]
Available at: <https://www.dofactory.com/sql/subquery>
[Accessed 8 January 2021].

Anon., n.d. *StackOverflow sort numbers maintaining original index*. [Online]
Available at: <https://stackoverflow.com/questions/65480072/is-there-a-way-to-sort-a-list-of-numbers-while-maintaining-their-original-index>
[Accessed 28 December 2020].

Anon., n.d. *The Checkbox Code*. [Online]
Available at: <https://www.homeandlearn.co.uk/NET/nets4p14.html>
[Accessed 4 January 2021].

Appendix

User Interface



StudentMenu

ACCOUNT ASSIGNMENTS MANUAL TEST AUTOMATIC TEST MY RESULTS

Test that generates intervals based on your previous performance

TAKE TEST

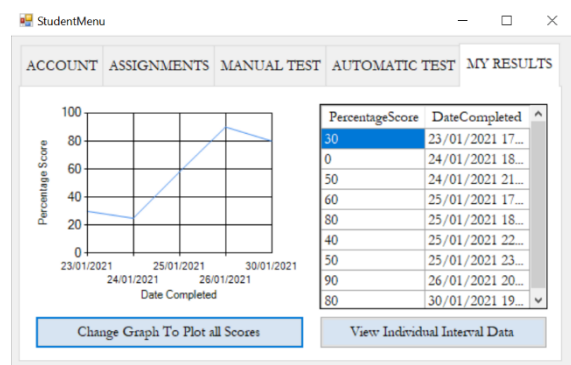
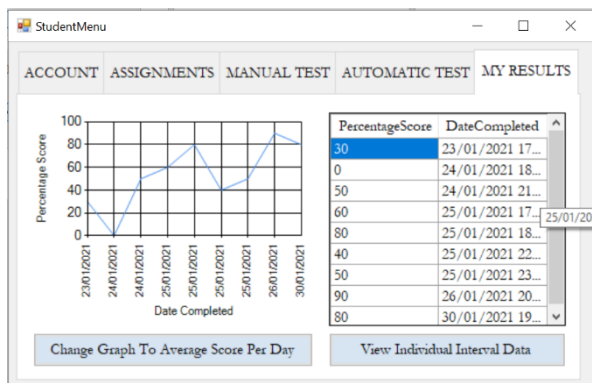
StudentMenu

ACCOUNT ASSIGNMENTS MANUAL TEST AUTOMATIC TEST MY RESULTS

Level 1: 2nds, 3rds, Perfect 4th, 5th and 8th
 Level 2: All Major, minor and perfect intervals
 Level 3: Major, minor, perfect, diminished and augmented intervals
 Level 4: All intervals and compound intervals

Level 2

TAKE TEST



IntervalData

INTERVAL DATA

Quality	Number	TotalTested	TotalCorrect	PercentageCorrect
Minor	2nd	6	6	100
Major	2nd	7	7	100
Minor	3rd	12	8	67
Major	3rd	5	3	60
Perfect	4th	8	3	38
Augmented	4th	1	0	0
Perfect	5th	4	4	100
Minor	6th	5	0	0
Major	6th	6	5	83

DisplayTestForm

Question Number 1

☐ Major
☒ Minor
☐ Perfect
☐ Augmented

☐ 2nd
☐ 3rd
☐ 4th
☐ 5th
☒ 6th
☐ 7th
☐ 8th
☐ 9th
☐ 10th

PLAY INTERVAL

SUBMIT ANSWER

DisplayTestResultsForm

RESULTS 50%

QuestionNumber	Quality	Number	Correct
3	Perfect	5th	<input checked="" type="checkbox"/>
4	Perfect	8th	<input type="checkbox"/>
5	Major	6th	<input type="checkbox"/>
6	Perfect	5th	<input checked="" type="checkbox"/>
7	Major	7th	<input type="checkbox"/>
8	Minor	2nd	<input checked="" type="checkbox"/>
9	Major	6th	<input checked="" type="checkbox"/>

TeacherMenu

ACCOUNT CLASSES MANUAL TEST AUTOMATIC TEST STUDENT RESULTS

Account Type Teacher

Username AAA

Name Amy

LOG OUT

Godalming College

TeacherMenu

ACCOUNT CLASSES MANUAL TEST AUTOMATIC TEST STUDENT RESULTS

ClassName	ClassID	NumberOfStudents
Amy's Class 1	213213	2
Amy's Class 3	272656	0
Amy's Class 2	333444	1
Amy's Class 4	389340	0

CREATE NEW CLASS

NewClass

Class Name:

CREATE CLASS

TeacherMenu

ACCOUNT CLASSES MANUAL TEST AUTOMATIC TEST STUDENT RESULTS

Select 10 Intervals To Test

- ☐ Minor 2nd
- ☐ Major 2nd
- ☐ Minor 3rd
- ☐ Major 3rd
- ☐ Perfect 4th
- ☐ Augmented 4th
- ☐ Perfect 5th
- ☐ Minor 6th
- ☐ Major 6th
- ☐ Minor 7th
- ☐ Major 7th

Select Class

Amy's Class 1

Select Due Date

15 March 2021

SET TEST

TeacherMenu

ACCOUNT CLASSES MANUAL TEST AUTOMATIC TEST STUDENT RESULTS

Level 1: 2nds, 3rds, Perfect 4th, 5th and 8th

Level 2: All Major, minor and perfect intervals

Level 3: Major, minor, perfect, diminished and augmented intervals

Level 4: All intervals and compound intervals

Level 1

Amy's Class 1

SET TEST

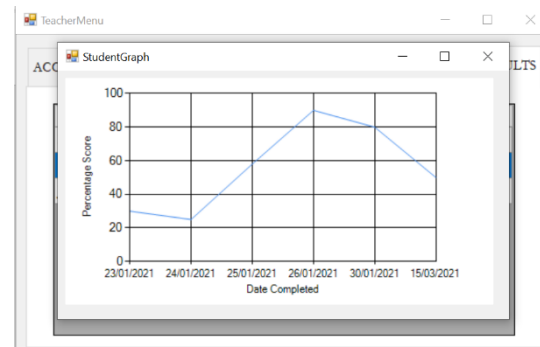
Due Date

15 March 2021

TeacherMenu

ACCOUNT CLASSES MANUAL TEST AUTOMATIC TEST STUDENT RESULTS

FirstName	Surname	StudentID	ClassName	AverageScore
Paul	Rodd	192267	Amy's Class 1	40
Katie	Taylor	192344	Amy's Class 1	53
Joel	Peters	196666	Amy's Class 2	17



Code

Launch Page

```
Public Class LaunchPage

    Private Sub btnLogIn_Click(sender As Object, e As EventArgs) Handles
btnLogIn.Click
        'opens log in form
        Me.Hide()
        LogInForm.Show()
    End Sub

    Private Sub btnCreateAccount_Click(sender As Object, e As EventArgs) Handles
btnCreateAccount.Click
        ' opens create account form
        Me.Hide()
        CreateAccountForm.Show()
    End Sub

End Class
```

Create Account

```
Public Class CreateAccountForm

    Private Sub btnStudentAccount_Click(sender As Object, e As EventArgs) Handles
btnStudentAccount.Click
        ' goes to student create account page
        Me.Close()
        CreateStudentAccount.Show()
    End Sub

    Private Sub btnTeacherAccount_Click(sender As Object, e As EventArgs) Handles
btnTeacherAccount.Click
        ' goes to teacher create account page
        Me.Close()
        CreateTeacherAccount.Show()
    End Sub

End Class
```

Create Student Account

```
Public Class CreateStudentAccount
    Private StudentID As String
    Private FirstName As String
    Private Surname As String
    Private Password As String
    Private ConfirmPassword As String
    Private EmptyBox As Boolean
    Private CorrectID As Boolean
    Private PasswordMatch As Boolean
    Private UniqueAccount As Boolean

    Private Sub btnConfirmCreateStudentAccount_Click(sender As Object, e As EventArgs)
Handles btnConfirmCreateStudentAccount.Click
        EmptyBox = False
    End Sub
End Class
```

```

UniqueAccount = False
PasswordMatch = True
CorrectID = True
' resets colour to white so red error boxes are accurate to this round of data
For Each c In Me.Controls
    If TypeName(c) = "TextBox" Then
        c.BackColor = Color.White
    End If
Next
' assigns information from textboxes to variables
StudentID = txtStudentNumberCreateStudentAccount.Text
Try
    FirstName = txtFirstNameCreateStudentAccount.Text.Substring(0, 1).ToUpper
+ txtFirstNameCreateStudentAccount.Text.Substring(1)
    Surname = txtSurnameCreateStudentAccount.Text.Substring(0, 1).ToUpper +
txtSurnameCreateStudentAccount.Text.Substring(1)
Catch
    EmptyBox = True
End Try
Password = txtPasswordCreateStudentAccount.Text
ConfirmPassword = txtConfirmPasswordCreateStudentAccount.Text
' validate textbox inputs
Validation()
' check to see if the account already exists
CheckAgainstDatabase()
' print error messages
PrintMessages()
' closes form if all data is entered correctly
If EmptyBox = False And CorrectID = True And PasswordMatch = True Then
    SaveStudentDetailsToDatabase()
    Me.Close()
    LogInForm.Show()
End If
End Sub

Private Sub Validation()
' checks to see if all textboxes have information in them and turns the
incorrect textboxes red
For Each c In Me.Controls
    If TypeName(c) = "TextBox" Then
        If c.Text = "" Then
            EmptyBox = True
            c.BackColor = Color.MistyRose
        End If
    End If
Next
' checks to see if studentID is the correct format
If StudentID.Length <> 6 Then
    CorrectID = False
    txtStudentNumberCreateStudentAccount.BackColor = Color.MistyRose
End If
If Not IsNumeric(StudentID) Then
    CorrectID = False
    txtStudentNumberCreateStudentAccount.BackColor = Color.MistyRose
End If
' checks to see if passwords match
If Password <> ConfirmPassword Then
    PasswordMatch = False
    txtConfirmPasswordCreateStudentAccount.BackColor = Color.MistyRose
End If
End Sub

```

```

Private Sub PrintMessages()
    ' print messages based on what is wrong with the inputs
    If EmptyBox = True Then
        MsgBox("Please fill in all boxes.")
    End If
    If CorrectID = False Then
        MsgBox("Your StudentID is in the wrong format, it must be 6 numbers.")
    End If
    If PasswordMatch = False Then
        MsgBox("Your passwords do not match.")
    End If
    If UniqueAccount = False Then
        MsgBox("There is already an account associated with this Student ID")
    End If
End Sub

Private Sub CheckAgainstDatabase()
    ' checks to see if the student account is already in the database
    Dim SQLString As String
    SQLString = "SELECT COUNT(StudentID) FROM tblStudent
                WHERE StudentID = @ID"
    Dim TeacherIDCheckDB As New DatabaseConnection(SQLString)
    If TeacherIDCheckDB.SelectStringID(StudentID) = "0" Then
        ' if it is not in database unique account is set to true
        UniqueAccount = True
    Else
        ' if it is in the database the textbox is set to red and unique account
remains false
        txbStudentNumberCreateStudentAccount.BackColor = Color.MistyRose
    End If
End Sub

Private Sub SaveStudentDetailsToDatabase()
    ' saves the details of the student account to the database
    Dim SQLString As String
    SQLString = "INSERT INTO tblStudent (StudentID, FirstName, Surname, Password)
VALUES (@ID, @FirstName ,@Surname, @Password)"
    Dim SaveStudentDB As New DatabaseConnection(SQLString)
    SaveStudentDB.SaveUser(StudentID, FirstName, Surname, Password)
    MsgBox("Account created - You many now log in.")
End Sub

Private Sub pbBackToMenu_Click(sender As Object, e As EventArgs) Handles
pbBackToMenu.Click
    ' back to launch page
    Me.Close()
    LaunchPage.Show()
End Sub
End Class

```

Create Teacher Account

```

Public Class CreateTeacherAccount
    ' -----HARD CODED FOR TESTING -----
    ' -----
    Private ListOfConfirmedTeacherIDs() As String = {"AAA", "ABC", "JAD", "PSC"}
    ' List is hard coded for testing but in the actual implementation of the program
    ' it would be stored in and read from a file
    ' -----
    Private TeacherID As String
    Private FirstName As String
    Private Surname As String
    Private Password As String
    Private ConfirmPassword As String
    Private EmptyBox As Boolean
    Private CorrectID As Boolean
    Private PasswordMatch As Boolean
    Private ConfirmedTeacher As Boolean
    Private UniqueAccount As Boolean

    Private Sub btnConfirmCreateTeacherAccount_Click(sender As Object, e As EventArgs)
Handles btnConfirmCreateTeacherAccount.Click
        EmptyBox = False
        UniqueAccount = False
        PasswordMatch = True
        ConfirmedTeacher = False
        CorrectID = False
        ' resets colour to white so red error boxes are accurate to this round of data
        For Each c In Me.Controls
            If TypeName(c) = "TextBox" Then
                c.BackColor = Color.White
            End If
        Next

        ' assigns information from textboxes to variables
        TeacherID = txtTeacherIDCreateTeacherAccount.Text.ToUpper
        Try
            FirstName = txtFirstNameCreateTeacherAccount.Text.Substring(0, 1).ToUpper
+ txtFirstNameCreateTeacherAccount.Text.Substring(1)
            Surname = txtSurnameCreateTeacherAccount.Text.Substring(0, 1).ToUpper +
txtSurnameCreateTeacherAccount.Text.Substring(1)
        Catch
            EmptyBox = True
        End Try
        Password = txtPasswordCreateTeacherAccount.Text
        ConfirmPassword = txtConfirmPasswordCreateTeacherAccount.Text

        ' validates inputs
        Validation()

        ' checks to see if account has been previously created
        CheckAgainstDatabase()

        ' prints error messages
        PrintMessages()

        ' closes form if all inputs are correct
        If EmptyBox = False And CorrectID = True And PasswordMatch = True And
UniqueAccount = True Then
            SaveTeacherDetailsToDatabase()
            Me.Close()
        End If
    End Sub
End Class

```

```

        LogInForm.Show()
    End If
End Sub

Private Sub SaveTeacherDetailsToDatabase()
    ' saves the new account to the database
    Dim SQLString As String
    SQLString = "INSERT INTO tblTeacher VALUES (@ID, @FirstName ,@Surname,
@Password)"
    Dim SaveTeacherDB As New DatabaseConnection(SQLString)
    SaveTeacherDB.SaveUser(TeacherID, FirstName, Surname, Password)
    MsgBox("Account created - You many now log in.")
End Sub

Private Sub CheckAgainstDatabase()
    ' checks to see if the account is already in the database
    Dim SQLString As String
    SQLString = "SELECT COUNT(TeacherID) FROM tblTeacher
        WHERE TeacherID = @ID"
    Dim TeacherIDCheckDB As New DatabaseConnection(SQLString)
    If TeacherIDCheckDB.SelectStringID(TeacherID) = "0" Then
        ' if it is not in database then unique account set to true
        UniqueAccount = True
    Else
        ' if the account has already been created then textbox is made red
        txbTeacherIDCreateTeacherAccount.BackColor = Color.MistyRose
    End If
End Sub

Private Sub Validation()
    ' checks to see if all textboxes have information in them and turns the
incorrect textboxes red
    For Each c In Me.Controls
        If TypeName(c) = "TextBox" Then
            If c.Text = "" Then
                EmptyBox = True
                c.BackColor = Color.MistyRose
            End If
        End If
    Next
    ' checks to see if teacherID is the correct format
    If TeacherID.Length <> 3 Then
        CorrectID = False
        txbTeacherIDCreateTeacherAccount.BackColor = Color.MistyRose
    End If
    Try
        TeacherID = CInt(TeacherID)
    Catch ex As Exception
        CorrectID = True
    End Try
    ' checks to see if teacherId is of a confirmed teacher (to prevent students
creating teacher accounts)
    For x = 0 To ListOfConfirmedTeacherIDs.Length - 1
        If TeacherID = ListOfConfirmedTeacherIDs(x) Then
            ConfirmedTeacher = True
        End If
    Next
    ' checks to see if passwords match
    If Password <> ConfirmPassword Then
        PasswordMatch = False
        txbConfirmPasswordCreateTeacherAccount.BackColor = Color.MistyRose
    End If

```

```

End Sub

Private Sub PrintMessages()
    ' prints the messages based on what is wrong with the data entered
    If EmptyBox = True Then
        MsgBox("Please fill in all boxes.")
    End If
    If CorrectID = False Then
        MsgBox("Your TeacherID is in the wrong format, it must be 3 letters.")
    End If
    If PasswordMatch = False Then
        MsgBox("Your passwords do not match.")
    End If
    If ConfirmedTeacher = False Then
        MsgBox("This is not a known Teacher ID.")
    End If
    If UniqueAccount = False Then
        MsgBox("There is already an account associated with this Teacher ID.")
    End If
End Sub

Private Sub pbBackToMenu_Click(sender As Object, e As EventArgs) Handles
pbBackToMenu.Click
    ' goes back to launch page
    Me.Close()
    LaunchPage.Show()
End Sub

End Class

```

Log In

```

Public Class LogInForm
    Private Shared CurrentID As String
    Private StudentAccount As Boolean
    Private Password As String
    Private AccountInDatabase As Boolean
    Private EmptyBoxes As Boolean
    Private CorrectID As Boolean

    Private Sub btnSubmitLoginDetails_Click(sender As Object, e As EventArgs) Handles
btnSubmitLoginDetails.Click
        AccountInDatabase = False
        EmptyBoxes = False
        CorrectID = True
        ' assign variables from textbox input
        CurrentID = txbUsernameLogin.Text
        Password = txbPasswordLogin.Text
        ' validate inputs
        Validation()
        ' check database to see if log in matches database
        If EmptyBoxes = False And CorrectID = True Then
            CheckDatabaseForLogin()
        End If
        ' display error messages
        DisplayMessages()
        If EmptyBoxes = False And CorrectID = True And AccountInDatabase = True Then
            If StudentAccount = True Then
                Me.Close()
                StudentMenu.Show()
            Else

```

```

        Me.Close()
        TeacherMenu.Show()
    End If
End If
End Sub

Private Sub Validation()
    ' presence check
    If CurrentID = "" Or Password = "" Then
        EmptyBoxes = True
    End If
    ' decides if currentID is student or teacher
    If CurrentID.Length = 6 Then
        StudentAccount = True
    ElseIf CurrentID.Length = 3 Then
        StudentAccount = False
    Else
        CorrectID = False
    End If
    ' type check for studentID and teacherID
    If StudentAccount = True Then
        If Not IsNumeric(CurrentID) Then
            CorrectID = False
        End If
    Else
        Try
            CurrentID = CInt(CurrentID)
        Catch ex As Exception
            CorrectID = True
            CurrentID = CurrentID.ToUpper
        End Try
    End If
End Sub

Private Sub CheckDatabaseForLogin()
    Dim SQLString As String
    ' use string length to determine if it's a student or teacher logging in
    If CurrentID.Length = 6 Then
        StudentAccount = True
        SQLString = "SELECT COUNT(StudentID) FROM tblStudent
                    WHERE StudentID = @Username AND Password = @Password"
    Else
        StudentAccount = False
        SQLString = "SELECT COUNT(TeacherID) FROM tblTeacher
                    WHERE TeacherID = @Username AND Password = @Password"
    End If
    ' search database for login info
    Dim SearchUserDB As New DatabaseConnection(SQLString)
    If SearchUserDB.SearchUser(CurrentID, Password) = 1 Then
        AccountInDatabase = True
    End If
End Sub

Private Sub DisplayMessages()
    ' displays error messages based on validation
    If EmptyBoxes = True Then
        MsgBox("Please fill in all boxes.")
    End If
    If AccountInDatabase = False And EmptyBoxes = False Then
        MsgBox("This information does not match an account. Please try again.")
    End If
    If CorrectID = False Then

```

```

        MsgBox("Your ID is in the wrong format.")
    End If
End Sub

Private Sub pbBackToMenu_Click(sender As Object, e As EventArgs) Handles
pbBackToMenu.Click
    ' goes back to launch page
    Me.Close()
    LaunchPage.Show()
End Sub

Public Function GetStudentAccount()
    If StudentAccount = True Then
        Return "Student"
    Else
        Return "Teacher"
    End If
End Function

Public Function GetCurrentID()
    Return CurrentID
End Function

End Class

```

Student Menu

```

Public Class StudentMenu
    Private SQLStr As String
    Private Shared ThisTest As TakeTest
    Private ClassID As String
    Private GraphMode As Boolean

    Private Sub StudentMenu_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        LoadStudentMenu()
    End Sub

    Private Sub LoadStudentMenu()
        ' displays account type, username and name for Account tab
        Dim LoginInformation As New LogInForm
        AccountTypeLabel.Text = "Student"
        UsernameLabel.Text = LogInForm.GetCurrentID
        SQLStr = "SELECT FirstName FROM tblStudent WHERE StudentID = @ID"
        Dim GetNameDB As New DatabaseConnection(SQLStr)
        NameLabel.Text = GetNameDB.SelectStringID(LogInForm.GetCurrentID)

        'display for assignment page
        SQLStr = "SELECT COUNT(Class) FROM tblStudent WHERE StudentID = @ID"
        Dim GetClassDB As New DatabaseConnection(SQLStr)
        ' sees if student is a member of a class
        If GetClassDB.SelectStringID(LogInForm.GetCurrentID) = "0" Then
            'lets student have the option to join a class
            ' shows and hides appropriate labels etc
            messagelabel.Text = "You are not currently part of a class.
            When you join a class your assignments will appear here."
            btnLeaveClass.Hide()
            btnJoinClass.Show()
            lblEnterClassID.Show()
            tbxClassID.Show()
            lblAssignmentsCompleted.Hide()
            dgvCompleteAssignments.Hide()
        End If
    End Sub
End Class

```



```

        dgvCurrentAssignments.Hide()
Else
    SQLStr = "SELECT Class FROM tblStudent WHERE StudentID = @ID"
    Dim GetClassIDDB As New DatabaseConnection(SQLStr)
    ClassID = GetClassIDDB.SelectStringID(LogInForm.GetCurrentID)
    'displays current and past assignments for the class
    ' shows and hides appropriate labels etc
    btnLeaveClass.Show()
    dgvCompleteAssignments.Show()
    dgvCurrentAssignments.Show()
    btnJoinClass.Hide()
    lblEnterClassID.Hide()
    tbxClassID.Hide()
    messagelabel.Text = "ASSIGNMENTS DUE:"
    lblAssignmentsCompleted.Text = "COMPLETED ASSIGNMENTS:"
    SetUpAssignments(ClassID)
End If

'display for my results page
'displays chart
GraphMode = True
ChangeGraph()
'displays grid
Dim DisplaySummaryTableDB As New DatabaseConnection(SQLStr)
DisplaySummaryTableDB.SummaryTable(LogInForm.GetCurrentID)
End Sub

Private Sub btnLogOut_Click(sender As Object, e As EventArgs) Handles btnLogOut.Click
    ' logs user out of account and goes back to launch page
    Me.Close()
    LaunchPage.Show()
    MsgBox("You have successfully logged out.")
End Sub

Private Sub btnSetManualTest_Click(sender As Object, e As EventArgs) Handles
btnSetManualTest.Click
    ' gets data from combo box and assigns value to level
    Dim LevelSelected As Boolean = True
    Dim Level As Integer
    If cbxAutoTestLevel.Text = "Level 1" Then
        Level = 1
    ElseIf cbxAutoTestLevel.Text = "Level 2" Then
        Level = 2
    ElseIf cbxAutoTestLevel.Text = "Level 3" Then
        Level = 3
    ElseIf cbxAutoTestLevel.Text = "Level 4" Then
        Level = 4
    Else
        MsgBox("Please select a level")
        LevelSelected = False
    End If
    ' generates test
    If LevelSelected Then
        Dim MyTest As New LevelTest(Level, True)
        ' take test
        ThisTest = New TakeTest(MyTest.GetTestID(), LogInForm.GetCurrentID)
        Me.Close()
        DisplayTestForm.Show()
    End If
End Sub

```

```

Private Sub btnTakeAutomaticTest_Click(sender As Object, e As EventArgs) Handles
btnTakeAutomaticTest.Click
    ' generates test
    Dim MyTest As New PersonalTest()
    ' take test
    ThisTest = New TakeTest(MyTest.GetTestID, LogInForm.GetCurrentID)
    ' display test
    Me.Close()
    DisplayTestForm.Show()
End Sub

Private Sub btnLeaveClass_Click(sender As Object, e As EventArgs) Handles
btnLeaveClass.Click
    ' sets class to null in the students record
    SQLStr = "UPDATE tblStudent SET Class = null WHERE StudentID = @ID"
    Dim DeleteClassDB As New DatabaseConnection(SQLStr)
    DeleteClassDB.DeleteFromStudent(LogInForm.GetCurrentID)
    MsgBox("You have been removed from the class.")
    ' reloads menu so they can reenter a new class code
    LoadStudentMenu()
End Sub

Private Sub btnJoinClass_Click(sender As Object, e As EventArgs) Handles
btnJoinClass.Click
    Dim ClassID As String = tbxClassID.Text
    Dim TestString As Integer
    Dim ValidFormat As Boolean = True
    ' validate that the input is 6 integers
    If ClassID.Length = 6 Then
        Try
            TestString = ClassID

        Catch
            ValidFormat = False
        End Try
    Else
        ValidFormat = False
    End If
    ' check to see if the class is in the database if it is a valid class ID
    If ValidFormat Then
        SQLStr = "SELECT COUNT(ClassID) FROM tblClass WHERE ClassID = @ID"
        Dim GetClassIDDB As New DatabaseConnection(SQLStr)
        If GetClassIDDB.SelectStringID(ClassID) = "0" Then
            ' the class ID is not in the database so return message
            MsgBox("Please enter a valid class code.")
        Else
            ' the class id is in the database so add to student account and set up
the list of assignments
            SQLStr = "UPDATE tblStudent SET Class = @ClassID WHERE StudentID =
@StudentID"
            Dim SaveClassDB As New DatabaseConnection(SQLStr)
            SaveClassDB.UpdateStudent(ClassID, LogInForm.GetCurrentID)
            LoadStudentMenu()
        End If
    Else
        ' return message if not a valid format
        MsgBox("Please enter a 6-digit class code.")
    End If
End Sub

Private Sub SetUpAssignments(ByVal ClassID As String)
    ' fills out due assignment table

```

```

        SQLStr = "SELECT DISTINCT TestName, ClassName, DueDate FROM tblAssignment LEFT
JOIN
    (SELECT TestID
    FROM tblResult
    WHERE StudentID = @StudentID)
    tblResult
    ON tblAssignment.TestID=tblresult.TestID
    INNER JOIN tblClass ON tblAssignment.ClassID = tblClass.ClassID
    WHERE tblResult.TestID is NULL AND tblAssignment.ClassID = @ClassID"
    Dim DisplayCurrentDB As New DatabaseConnection(SQLStr)
    DisplayCurrentDB.AssignmentsTable(ClassID, LogInForm.GetCurrentID)

    ' fills out completed assignment table
    SQLStr = "SELECT DISTINCT TestName, ClassName, DueDate FROM tblAssignment LEFT
JOIN
    (SELECT TestID
    FROM tblResult
    WHERE StudentID = @StudentID)
    tblResult
    ON tblAssignment.TestID=tblresult.TestID
    INNER JOIN tblClass ON tblAssignment.ClassID = tblClass.ClassID
    WHERE tblResult.TestID IS NOT NULL AND tblAssignment.ClassID = @ClassID"
    Dim DisplayCompleteDB As New DatabaseConnection(SQLStr)
    DisplayCompleteDB.CompleteAssignmentsTable(ClassID, LogInForm.GetCurrentID)
End Sub

Private Sub btnViewIntervalData_Click(sender As Object, e As EventArgs) Handles
btnViewIntervalData.Click
    ' returns the interval name, number of times it has been guessed correctly,
    number of times it has been tested and percentage accuracy
    SQLStr = "SELECT Quality, Number, SUM(Correct = True OR Correct = False) As
TotalTested, SUM(Correct = True) As TotalCorrect, CAST(SUM(Correct = True) /
SUM(Correct = True OR Correct = False) * 100 As int) As PercentageCorrect
    FROM tblInterval
    INNER JOIN tblQuestion ON tblinterval.IntervalID = tblquestion.IntervalID
    INNER JOIN tblresult ON tblquestion.QuestionNumber = tblresult.QuestionNumber
AND tblquestion.TestID = tblresult.TestID
    WHERE StudentID = @StudentID
    GROUP BY Quality, Number
    ORDER BY tblInterval.IntervalID ASC"
    Dim DisplayIntervalTableDB As New DatabaseConnection(SQLStr)
    DisplayIntervalTableDB.IntervalTable(LogInForm.GetCurrentID)
    Me.Hide()
    IntervalData.Show()
End Sub

Private Sub btnGraphChange_Click(sender As Object, e As EventArgs) Handles
btnGraphChange.Click
    ' changes graph to opposite mode to what it currently is
    ChangeGraph()
End Sub

Private Sub ChangeGraph()
    If GraphMode = True Then
        ' creates graph will all data points plotted
        btnGraphChange.Text = "Change Graph To Average Score Per Day"
        SQLStr = "SELECT PercentageScore, DateCompleted FROM tblFinishedTest WHERE
StudentID = @StudentID ORDER BY DateCompleted ASC"
    Else
        ' creates graph with an average for each day plotted
        btnGraphChange.Text = "Change Graph To Plot all Scores"
    End If
End Sub

```

```

        SQLStr = "SELECT AVG(PercentageScore) As PercentageScore, DateCompleted
FROM tblfinishedtest
    WHERE StudentID = @StudentID
    GROUP BY CAST(DateCompleted AS DATE)
    ORDER BY DateCompleted ASC"
    End If
    ' displays the graph
    Dim DisplaySummaryChartDB As New DatabaseConnection(SQLStr)
    DisplaySummaryChartDB.SummaryChart(LogInForm.GetCurrentID)
    crtSummaryData.ChartAreas("ChartArea1").AxisX.Title = "Date Completed"
    crtSummaryData.ChartAreas("ChartArea1").AxisY.Title = "Percentage Score"
    crtSummaryData.ChartAreas("ChartArea1").AxisX.IsMarginVisible = False
    crtSummaryData.ChartAreas("ChartArea1").AxisY.Maximum = 100
    GraphMode = Not GraphMode
End Sub

Private Sub dgvCurrentAssignments_ClickSelectedCell(ByVal sender As System.Object,
ByVal e As System.Windows.Forms.DataGridViewCellEventArgs) Handles
dgvCurrentAssignments.CellContentClick
    Dim TestID As String
    Dim TestName As String
    Dim DueDate As Date
    ' gets the testname and due date of assignment that was clicked
    Try
        TestName = dgvCurrentAssignments.Rows(e.RowIndex).Cells(0).Value
        DueDate = dgvCurrentAssignments.Rows(e.RowIndex).Cells(2).Value
        ' asks user if they want to take the test
        Dim Answer As Integer = MsgBox("Do you want to take " & TestName & " ?",
vbQuestion + vbYesNo + vbDefaultButton2, "Message Box Title")
        If Answer = vbYes Then
            ' gets the TestID for the selected test
            SQLStr = "SELECT TestID FROM tblAssignment WHERE TestName = @TestName
AND DueDate = @DueDate AND ClassID = @ClassID"
            Dim GetTestID As New DatabaseConnection(SQLStr)
            TestID = (GetTestID.SelectThreeStrings(TestName, DueDate,
ClassID)).ToString
            ' takes test
            ThisTest = New TakeTest(TestID, LogInForm.GetCurrentID)
            Me.Close()
            DisplayTestForm.Show()
        End If
    Catch
        ' prevents system from crashing when a heading is clicked
    End Try
End Sub

Public Function GetThisTest()
    Return ThisTest
End Function

Private Sub pbRefreshAssignments_Click(sender As Object, e As EventArgs) Handles
pbRefreshAssignments.Click
    ' refreshes assignment page (if teacher sets test whilst they are logged in
they can take it without having to log out then back in)
    LoadStudentMenu()
End Sub
End Class

```

Display Test

```

Public Class DisplayTestForm
    Private QuestionCounter As Integer
    Private CurrentTest As TakeTest = StudentMenu.GetThisTest()
    Private TestID As String = CurrentTest.GetTestID
    Private TimesPlayed As Integer
    Private SQLStr As String
    Private Results(9) As Boolean
    ' ----- HARD CODED FOR TESTING -----
    Private MaxTimesPlayed As Integer = 2
    ' Number of times an interval can be played is hard coded for testing purposes but
    ' it actual implementation would be read from a file
    ' -----
    Private Shared ThisTestAverageScore As Integer

    Private Sub DisplayTestForm_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
        ' initialises the test counter and writes the question number label
        QuestionCounter = 1
        lblQuestion.Text = "Question Number " & QuestionCounter
    End Sub

    Private Sub btnPlayInterval_Click(sender As Object, e As EventArgs) Handles
btnPlayInterval.Click
        ' plays interval when button is clicked - only allowed to play interval twice
        ' per question
        If TimesPlayed < MaxTimesPlayed Then
            CurrentTest.PlayInterval(QuestionCounter)
            TimesPlayed += 1
            If TimesPlayed = MaxTimesPlayed Then
                btnPlayInterval.Enabled = False
                btnPlayInterval.BackColor = Color.DarkGray
            End If
        End If
    End Sub

    Private Sub btnConfirmAnswer_Click(sender As Object, e As EventArgs) Handles
btnConfirmAnswer.Click
        ' checks to see if one quality and one number have been selected as the
        ' answer, if not gives error message
        If CheckedListBox1.CheckedIndices.Count = 1 And
        CheckedListBox2.CheckedIndices.Count = 1 Then
            MarkAnswer()
        Else
            MsgBox("Please select one quality and one number")
        End If
    End Sub

    Private Sub MarkAnswer()
        Dim Quality As String
        Dim Number As String
        ' calls database and finds the quality from the question number and testID
        SQLStr = "SELECT Quality FROM tblInterval INNER JOIN tblQuestion ON
tblInterval.IntervalID = tblQuestion.IntervalID WHERE QuestionNumber = " &
QuestionCounter & " AND TestID = @ID"
        Dim SelectQuality As New DatabaseConnection(SQLStr)
        Quality = SelectQuality.SelectStringID(TestID)
        ' calls database and finds the number from the question number and testID
        SQLStr = "SELECT Number FROM tblInterval INNER JOIN tblQuestion ON
tblInterval.IntervalID = tblQuestion.IntervalID WHERE QuestionNumber = " &
QuestionCounter & " AND TestID = @ID"
        Dim SelectNumber As New DatabaseConnection(SQLStr)

```

```

Number = SelectNumber.SelectStringID(TestID)
' checks to see if the data in the checked boxes matches the correct answer
If CheckedListBox1.CheckedItems(0) = Quality And
CheckedListBox2.CheckedItems(0) = Number Then
    MsgBox("Correct")
    ' adds boolean to results array
    Results(QuestionCounter - 1) = True
Else
    MsgBox("Incorrect: The correct answer was " & Quality & " " & Number)
    ' adds boolean to results array
    Results(QuestionCounter - 1) = False
End If
' goes to the next question if question counter is under 10
If QuestionCounter < 10 Then
    NextQuestion()
Else
    ' saves results to database and displays them after the 10th question
    SaveAndDisplayResults()
End If
End Sub

Private Sub NextQuestion()
    ' advances to the next question - increments question counter and resets
timesplayed and resets buttons and checkboxes
    QuestionCounter += 1
    lblQuestion.Text = "Question Number " & QuestionCounter
    TimesPlayed = 0
    btnPlayInterval.Enabled = True
    btnPlayInterval.BackColor = SystemColors.Control
    For i As Integer = 0 To CheckedListBox1.Items.Count - 1
        CheckedListBox1.SetItemChecked(i, False)
        CheckedListBox1.SetSelected(i, False)
    Next
    For i As Integer = 0 To CheckedListBox2.Items.Count - 1
        CheckedListBox2.SetItemChecked(i, False)
        CheckedListBox2.SetSelected(i, False)
    Next
End Sub

Private Sub SaveAndDisplayResults()
    ' save results to database
    For x = 1 To 10
        SQLStr = "INSERT INTO tblResult VALUES (@TestID, @StudentID,
@QuestionNumber, @Correct)"
        Dim SaveToResultsDB As New DatabaseConnection(SQLStr)
        SaveToResultsDB.SaveResults(TestID, LogInForm.GetCurrentID, x, Results(x -
1))
    Next
    ' saves results to summary database
    Dim TotalCorrect As Integer
    Dim TotalQuestions As Integer = 10
    Dim AverageScore As Integer
    ' counts the total answers correct

    For x = 0 To 9
        If Results(x) = True Then
            TotalCorrect += 1
        End If
    Next
    ThisTestAverageScore = TotalCorrect / TotalQuestions * 100

```

```

        ' saves to finishedtest table
        SQLStr = "INSERT INTO tblFinishedTest VALUES (@TestID, @StudentID,
@PercentageScore, @DateCompleted)"
        Dim SaveToFinishedTestDB As New DatabaseConnection(SQLStr)
        SaveToFinishedTestDB.SaveFinishedTest(TestID, LogInForm.GetCurrentID,
ThisTestAverageScore, DateTime.Now)

        ' creates new summary record if the student doesn't have one, if not it adds
to the existing one
        SQLStr = "SELECT COUNT(StudentID) FROM tblSummary WHERE StudentID = @ID"
        Dim FindStudentDB As New DatabaseConnection(SQLStr)
        If FindStudentDB.SelectStringID(LogInForm.GetCurrentID) = "0" Then
            ' saves to summary table
            SQLStr = "INSERT INTO tblSummary VALUES (@StudentID, @TotalQuestions,
@TotalCorrect, @AverageScore)"
            Dim SaveToSummaryDB As New DatabaseConnection(SQLStr)
            SaveToSummaryDB.SaveSummary(LogInForm.GetCurrentID, TotalQuestions,
TotalCorrect, ThisTestAverageScore)
        Else
            ' gets the old total questions and correct from the database and adds them
to the ones from this test and saves new values to database
            SQLStr = "SELECT TotalQuestions FROM tblSummary WHERE StudentID = @ID"
            Dim SelectTotalQs As New DatabaseConnection(SQLStr)
            TotalQuestions += SelectTotalQs.SelectStringID(LogInForm.GetCurrentID)
            SQLStr = "SELECT TotalCorrect FROM tblSummary WHERE StudentID = @ID"
            Dim SelectTotalCorrect As New DatabaseConnection(SQLStr)
            TotalCorrect += SelectTotalCorrect.SelectStringID(LogInForm.GetCurrentID)
            AverageScore = TotalCorrect / TotalQuestions * 100
            SQLStr = "UPDATE tblSummary SET TotalQuestions = @TotalQuestions,
TotalCorrect = @TotalCorrect, AverageScore = @AverageScore WHERE StudentID =
@StudentID"
            Dim UpdateSummaryDB As New DatabaseConnection(SQLStr)
            UpdateSummaryDB.SaveSummary(LogInForm.GetCurrentID, TotalQuestions,
TotalCorrect, AverageScore)
        End If

        ' display results
        SQLStr = "SELECT tblResult.QuestionNumber, Quality, Number, Correct FROM
tblResult
INNER JOIN tblQuestion ON tblResult.TestID = tblQuestion.TestID
AND tblresult.QuestionNumber = tblquestion.QuestionNumber
INNER JOIN tblInterval ON tblquestion.IntervalID = tblinterval.IntervalID
WHERE tblResult.TestID = @TestID AND StudentID = @StudentID"
        Dim DisplayResultsDB As New DatabaseConnection(SQLStr)
        DisplayResultsDB.ResultsTable(TestID, LogInForm.GetCurrentID)
        Me.Close()
        DisplayTestResultsForm.Show()
    End Sub

    Public Function GetThisTestAverageScore() As Integer
        Return ThisTestAverageScore
    End Function

End Class

```

Display Test Results

```
Public Class DisplayTestResultsForm
```

```
    Private Sub pbBackToMenu_Click(sender As Object, e As EventArgs) Handles
pbBackToMenu.Click
        ' goes back to student menu
        Me.Close()
        StudentMenu.Show()
    End Sub
```

```
    Private Sub DisplayTestResultsForm_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
        ' puts the percentage score from the test as a label on the form
        lblPercentageScore.Text = DisplayTestForm.GetThisTestAverageScore & "%"
    End Sub
End Class
```

Interval Page

```
Public Class IntervalData
```

```
    Private Sub pbBackToMenu_Click(sender As Object, e As EventArgs) Handles
pbBackToMenu.Click
        ' goes back to student menu
        Me.Close()
        StudentMenu.Show()
    End Sub
End Class
```

Student Graph

```
Public Class StudentGraph
```

```
    Private Sub StudentGraph_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
        ' creates graph with average per day data for the student selected
        Dim SQLStr As String
        SQLStr = "SELECT AVG(PercentageScore) As PercentageScore, DateCompleted FROM
tblfinishedtest
                WHERE StudentID = @StudentID
                GROUP BY CAST(DateCompleted AS DATE)
                ORDER BY DateCompleted ASC"
        Dim DisplayStudentChartDB As New DatabaseConnection(SQLStr)
        DisplayStudentChartDB.StudentChart(TeacherMenu.GetStudentID)
        crtStudentData.ChartAreas("ChartArea1").AxisX.Title = "Date Completed"
        crtStudentData.ChartAreas("ChartArea1").AxisY.Title = "Percentage Score"
        crtStudentData.ChartAreas("ChartArea1").AxisX.IsMarginVisible = False
        crtStudentData.ChartAreas("ChartArea1").AxisY.Maximum = 100
    End Sub
End Class
```

Teacher Menu

```
Public Class TeacherMenu
```

```
    Private SQLStr As String
    Private ClassName As String
    Private Shared ClassID As String
    Private Shared TestName As String
    Private Shared StudentID As String
    Private Shared DateDue As Date
    Private Shared Intervals(9) As String
```

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



```

        ' loads teacher menu
        LoadTeacherMenu()
    End Sub

    Private Sub LoadTeacherMenu()

        ' displays account type, username and name for Account tab
        Dim LoginInformation As New LogInForm
        AccountTypeLabel.Text = "Teacher"
        UsernameLabel.Text = LogInForm.GetCurrentID
        SQLStr = "SELECT FirstName FROM tblTeacher WHERE TeacherID = @ID"
        Dim GetNameDB As New DatabaseConnection(SQLStr)
        NameLabel.Text = GetNameDB.SelectStringID(LogInForm.GetCurrentID)

        'sets minimum date for today for due date setters
        DateTimePicker1.MinDate = Now.Date
        DateTimePicker2.MinDate = Now.Date

        ' display for automatic and manual test tab - adds list of classes to combo
select boxes
        LoadClasses()

        ' displays classes in class tab
        DisplayClasses()

        'displays student results tab
        DisplayStudentResults()
    End Sub

    Private Sub btnLogOut_Click(sender As Object, e As EventArgs) Handles
btnLogOut.Click
        ' logs user out of account and goes back to launch page
        Me.Close()
        LaunchPage.Show()
        MsgBox("You have successfully logged out.")
    End Sub

    Private Sub btnSetAutoTest_Click(sender As Object, e As EventArgs) Handles
btnSetAutoTest.Click
        ' gets data from combo box and assigns value to level
        Dim LevelSelected As Boolean = True
        Dim Level As Integer
        If cbxAutoTestLevel.Text = "Level 1" Then
            Level = 1
        ElseIf cbxAutoTestLevel.Text = "Level 2" Then
            Level = 2
        ElseIf cbxAutoTestLevel.Text = "Level 3" Then
            Level = 3
        ElseIf cbxAutoTestLevel.Text = "Level 4" Then
            Level = 4
        Else
            MsgBox("Please select a level")
            LevelSelected = False
        End If
        ' assigns value to classname, duedate and classID
        If LevelSelected Then
            ClassName = cbxAutoTestClass.Text
            DateDue = DateTimePicker1.Text
            SQLStr = "SELECT ClassID FROM tblClass WHERE ClassName = @ID"
            Dim GetClassIDDB As New DatabaseConnection(SQLStr)
            ClassID = GetClassIDDB.SelectStringID(ClassName)
            ' gets test name

```

```

        ValidateTestName()
        ' variables used to generate a new test
        Dim MyTest As New LevelTest(Level, False)
        ' outputs message
        MsgBox("Test Set")
    End If
End Sub

Private Sub btnSetTestManual_Click(sender As Object, e As EventArgs) Handles
btnSetTestManual.Click
    Dim count As Integer = 0
    ' validates inputs
    If ValidateInputs() = True Then
        'sets class name from combobox
        ClassName = cbxSelectClassManual.Text
        SQLStr = "SELECT ClassID FROM tblClass WHERE ClassName = @ID"
        Dim GetClassIDDB As New DatabaseConnection(SQLStr)
        ' gets ClassID from classname from database
        ClassID = GetClassIDDB.SelectStringID(ClassName)
        ' sets due date from date picker
        DateDue = DateTimePicker2.Text
        ' assigns the index of each checked box to an array which becomes interval
indexes
        For Each index In CheckedListBox1.CheckedIndices
            Intervals(count) = index
            count += 1
        Next
        ' gets test name
        ValidateTestName()
        ' creates a test :)
        Dim ThisTest As New Test(0, False)
        ' outputs message
        MsgBox("Test Set")
    End If
End Sub

Private Function ValidateInputs()
    ' checks to see that all info has been provided
    Dim ValidInput = True
    If CheckBoxesCount() = False Then
        MsgBox("Please select 10 intervals")
        ValidInput = False
    End If
    If cbxSelectClassManual.Text = " " Then
        MsgBox("Please select a class")
        ValidInput = False
    End If
    Return ValidInput
End Function

Private Function CheckBoxesCount()
    ' counts the number of checkboxes checked - needs to be 10 to be valid
    Dim count As Integer
    count = CheckedListBox1.CheckedIndices.Count
    If count = 10 Then
        Return True
    Else
        Return False
    End If
End Function

Private Sub ValidateTestName()

```

```

Dim UniqueName As Boolean
Do
    UniqueName = True
    ' prompts user to enter a test name
    TestName = InputBox("Enter test name: ", "Test Name", "")
    ' must be longer than 0 characters
    If TestName.Length = 0 Then
        MsgBox("Please enter a valid test name.")
        UniqueName = False
    Else
        ' checks to see if a test with that name, class and due date already
exists (not allowed to avoid confusion)
        SQLStr = "SELECT COUNT(TestName) FROM tblAssignment WHERE ClassID =
@ClassID AND DueDate = @DueDate AND TestName = @TestName"
        Dim GetUniqueNameDB As New DatabaseConnection(SQLStr)
        If GetUniqueNameDB.SelectThreeStrings(TestName, DateDue, ClassID) > 0
Then
            UniqueName = False
            MsgBox("You have already set a test with the same name and due
date to this class. Please try a new name.")
            End If
        End If
        ' loops until teacher enters a non-zero name that isnt in database
    Loop Until UniqueName
End Sub

Public Sub DisplayClasses()
    ' gets classID and className and number of students and display them on class
tab
    SQLStr = "SELECT ClassName, ClassID, COUNT(StudentID) As NumberOfStudents FROM
tblclass
LEFT JOIN tblStudent ON tblclass.ClassID = tblstudent.Class
WHERE TeacherID = @ID
GROUP BY ClassID"
    Dim DisplayClassDB As New DatabaseConnection(SQLStr)
    DisplayClassDB.ClassTable(LogInForm.GetCurrentID)
End Sub

Private Sub DisplayStudentResults()
    ' get student info and display on results tab
    SQLStr = "SELECT FirstName, Surname, tblStudent.StudentID, ClassName,
AverageScore FROM tblstudent
LEFT JOIN tblsummary ON tblstudent.StudentID = tblsummary.StudentID
INNER JOIN tblClass ON tblstudent.Class = tblClass.ClassID
WHERE TeacherID = @TeacherID"
    Dim DisplayStudentDB As New DatabaseConnection(SQLStr)
    DisplayStudentDB.StudentTable(LogInForm.GetCurrentID)
End Sub

Private Sub btnCreateNewClass_Click(sender As Object, e As EventArgs) Handles
btnCreateNewClass.Click
    ' opens new class form
    Me.Close()
    NewClass.Show()
End Sub

Public Sub LoadClasses()
    ' outputs all of the teacher's classes to the comboboxes
    Dim ClassNames As List(Of String)
    SQLStr = "SELECT ClassName FROM tblClass WHERE TeacherID = @ID"
    Dim ClassListDB As New DatabaseConnection(SQLStr)
    ClassNames = ClassListDB.GetClassList(LogInForm.GetCurrentID)

```

```

    If ClassNames.Count = 0 Then
        cbxAutoTestClass.Dispose()
        cbxSelectClassManual.Dispose()
        ' grey out button and replace combo box with label saying that there are
no classes
        btnSetAutoTest.Enabled = False
        btnSetTestManual.Enabled = False
        lblAutoNoClass.Text = "You currently have no classes to assign tests to.
In order to set tests please create a class"
        lblManualNoClass.Text = "You currently have no classes to assign tests to.
In order to set tests please create a class"
    Else
        ' assigns class list to combo boxes
        Dim comboSource As New Dictionary(Of String, String)()
        For x = 0 To ClassNames.Count - 1
            comboSource.Add(x + 1, ClassNames(x))
        Next
        cbxAutoTestClass.DataSource = New BindingSource(comboSource, Nothing)
        cbxAutoTestClass.DisplayMember = "Value"
        cbxAutoTestClass.ValueMember = "Key"
        cbxSelectClassManual.DataSource = New BindingSource(comboSource, Nothing)
        cbxSelectClassManual.DisplayMember = "Value"
        cbxSelectClassManual.ValueMember = "Key"
    End If
End Sub

Private Sub dgvStudentResults_ClickSelectedCell(ByVal sender As System.Object,
ByVal e As System.Windows.Forms.DataGridViewCellEventArgs) Handles
dgvStudentResults.CellContentClick
    ' gets data from the row that was clicked
    Try
        ' checks to see value of the "average score" column - if it is null then
they havent taken any tests and therefore dont have a graph
        If dgvStudentResults.Rows(e.RowIndex).Cells(4).Value Is DBNull.Value Then
            MsgBox("No progress data available.")
        Else
            ' assigns value to studentID from the row clicked
            StudentID = dgvStudentResults.Rows(e.RowIndex).Cells(2).Value
            ' shows that student's graph
            StudentGraph.Show()
        End If
    Catch
        ' prevents crash if a header gets clicked :)
    End Try
End Sub

' getters:

Public Function GetClassID()
    Return ClassID
End Function

Public Function GetDateDue()
    Return DateDue
End Function

Public Function GetIntervals()
    Return Intervals
End Function

Public Function GetTestName()
    Return TestName

```

```

End Function

Public Function GetStudentID()
    Return StudentID
End Function
End Class

New Class
Public Class NewClass
    Private SQLStr As String
    Private ClassName As String
    Private ClassID As String

    Private Sub btnCreateNewClass_Click(sender As Object, e As EventArgs) Handles
btnCreateNewClass.Click
        Dim validName As Boolean = False
        Dim validClassID As Boolean = False
        Dim randomNumber As String
        ' checks to see if a value has been added to the class name textbox
        If tbxCreateNewClassName.Text.Length < 1 Then
            MsgBox("Please enter a class name")
        Else
            ' checks to see if the teacher has another class with the same name
            SQLStr = "SELECT COUNT(ClassName) FROM tblClass WHERE ClassName = @ID"
            Dim GetClassNameDB As New DatabaseConnection(SQLStr)
            If GetClassNameDB.SelectStringID(tbxCreateNewClassName.Text) = 0 Then
                ClassName = tbxCreateNewClassName.Text
                validName = True
            Else
                MsgBox("Please enter a unique class name")
            End If
        End If
        ' generates a random 6 digit number to be the class code and then checks to
        see if it is already a class code
        If validName = True Then
            SQLStr = "SELECT COUNT(ClassID) FROM tblClass WHERE ClassID = @ID"
            Dim GetClassIDDB As New DatabaseConnection(SQLStr)
            Do
                randomNumber = RandomSixDigitNumber()
                If GetClassIDDB.SelectStringID(randomNumber) = "0" And
randomNumber.Length = 6 Then
                    validClassID = True
                End If
            Loop Until validClassID
            ' sets class ID
            ClassID = randomNumber
            ' saves new class to database
            SQLStr = "INSERT INTO tblClass VALUES (@ClassID, @ClassName, @TeacherID)"
            Dim SaveClassDB As New DatabaseConnection(SQLStr)
            SaveClassDB.SaveClass(ClassID, ClassName, LogInForm.GetCurrentID)
            MsgBox("Class created :)")
            ' goes back to teacher menu and reloads class list to show new class
            Me.Hide()
            Dim ReDisplay As New TeacherMenu
            TeacherMenu.Show()
            ReDisplay.DisplayClasses()
        End If
    End Sub

    Private Function RandomSixDigitNumber()
        Randomize()
        Return Int((1000000 * Rnd()) + 1)
    End Function
End Class

```

```
End Function
End Class
```

Question Class

```
Imports MySql.Data.MySqlClient
Imports System.Threading
```

Class Question

```
' 10 new objects created for each test: each have these properties
Property TestID As String
Property IntervalIndex As Integer
Property QuestionNumber As Integer
Property StartingNote As Integer
Property Direction As String

Sub New(ByVal ID As String, ByVal Index As Integer, ByVal Question As Integer,
ByVal Starting As Integer, ByVal AscOrDesc As String)
    TestID = ID
    IntervalIndex = Index
    QuestionNumber = Question
    StartingNote = Starting
    Direction = AscOrDesc
End Sub
```

```
End Class
```

Test Class

Class Test

```
' parent class to leveltest and personaltest
Protected studentUser As Boolean
Private ThisTest As New List(Of Question)
Private TestID As String
Protected Level As Integer
Protected Intervals(9) As Integer

Sub New(ByVal TestLevel As Integer, ByVal StudentOrTeacher As Boolean)
    Level = TestLevel
    studentUser = StudentOrTeacher
    GenerateTestID()
    GenerateIntervals()
    GenerateStartingNoteAndDirection()
    SaveToDatabase()
End Sub

Private Sub GenerateStartingNoteAndDirection()
    Dim SQLString As String
    Dim semitones As Integer
    Dim intervalIndex As Integer
    Dim startingNote As Integer
    Dim Direction As String
    ' loops for each of the 10 intervals in the test
    For n = 0 To 9
        intervalIndex = Intervals(n)
        ' makes call to database to find the number of semitones in the interval
        for the current question
        SQLString = "SELECT Semitones FROM tblInterval WHERE IntervalID =
@IntervalID"
        Dim SemitonesDB As New DatabaseConnection(SQLString)
        semitones = SemitonesDB.SelectIntegerID(intervalIndex)
```

```

        ' uses semitones in the interval to determine which starting notes are
possible
        ' (as if it was entirely random it might not be possible to play the
ending note as there is not an infinite number of notes)
        If semitones > 18 Then
            ' there are 36 notes available and therefore if the number of
semitones is above 18 the starting note cannot be in the middle of the range of notes
            ' randomly generates a starting note between 0 and 11 or 23 and 35
            If GenerateRandomNumber(0, 100) Mod 2 = 0 Then
                startingNote = GenerateRandomNumber(0, 11)
            Else
                startingNote = GenerateRandomNumber(23, 35)
            End If
        Else
            ' if semitones is less than 18 then the starting note can be any of
the 36
            startingNote = GenerateRandomNumber(0, 35)
        End If
        ' determines the direction by ensuring the ending note isn't out of the
range of notes
        If startingNote + semitones > 35 Then
            Direction = "Desc"
        ElseIf startingNote - semitones < 0 Then
            Direction = "Asc"
        Else
            ' if the direction will not cause the note to go out of range it is
randomised
            If GenerateRandomNumber(0, 100) Mod 2 = 0 Then
                Direction = "Asc"
            Else
                Direction = "Desc"
            End If
        End If
        ' adds all the information for this question to a new question in the list
for this test
        ThisTest.Add(New Question(TestID, intervalIndex, n + 1, startingNote,
Direction))
    Next
End Sub

Private Sub GenerateTestID()
    ' generates a unique string to be used as testID
    TestID = Guid.NewGuid.ToString("N")
End Sub

Protected Function GenerateRandomNumber(ByVal min As Integer, ByVal max As
Integer)
    ' generates a random number between two given values
    Randomize()
    Return Int((max - min + 1) * Rnd()) + min
End Function

Private Sub SaveToDatabase()
    Dim ClassID As String = TeacherMenu.GetClassID
    Dim DueDate As Date = TeacherMenu.GetDateDue
    Dim SQLString As String
    ' saves the information for each question in the test to the Question table
    For Each Question In ThisTest
        SQLString = "INSERT INTO tblQuestion VALUES (@TestID, @QuestionNumber
,@IntervalID, @StartingNote, @Direction)"
        Dim SaveQuestionsDB As New DatabaseConnection(SQLString)
    Next
End Sub

```

```

        SaveQuestionsDB.SaveQuestions(Question.TestID, Question.QuestionNumber,
        Question.IntervalIndex, Question.StartingNote, Question.Direction)
    Next
    ' if it is a teacher then the class it is assigned to and the due date for the
    test are saved to the Test table
    If studentUser = False Then
        SQLString = "INSERT INTO tblAssignment VALUES (@TestID, @TestName,
        @ClassID, @DueDate)"
        Dim SaveTestDB As New DatabaseConnection(SQLString)
        SaveTestDB.SaveAssignment(TeacherMenu.GetTestName, TestID, ClassID,
        DueDate)
    End If
End Sub

Public Function GetTestID()
    Return TestID
End Function

Overridable Sub GenerateIntervals()
    ' assigns the interval indexes of the checked boxes to the interval array
    For n = 0 To 9
        Intervals(n) = TeacherMenu.GetIntervals(n)
    Next
End Sub

End Class

```

Level Test Class

```

Class LevelTest
    Inherits Test
    Sub New(ByVal TestLevel As Integer, ByVal StudentUser As Boolean)
        ' runs when a new level test is created
        MyBase.New(TestLevel, StudentUser)
    End Sub

    Overrides Sub GenerateIntervals()
        Dim randomNumber As Integer
        'loops 10 times - an interval for each question.
        'Random Number Is indicative of the interval index: the interval indexes that
        can be used Is dependent on the level selected by the user
        For n = 0 To 9
            ' easiest intervals (2nds, 3rds, perfect 4ths, 5ths and 8ths)
            If Level = 1 Then
                Dim Random As Integer = GenerateRandomNumber(0, 6)
                If Random < 5 Then
                    Intervals(n) = GenerateRandomNumber(0, 4)
                ElseIf Random = 5 Then
                    Intervals(n) = 6
                ElseIf Random = 6 Then
                    Intervals(n) = 11
                End If
            End If
            ' intervals that are NOT compound or augmented
            If Level = 2 Then
                Do
                    randomNumber = GenerateRandomNumber(0, 11)
                Loop Until randomNumber <> 5
                Intervals(n) = randomNumber
            End If
            ' intervals that are NOT compound

```



```

        If Level = 3 Then
            Intervals(n) = GenerateRandomNumber(0, 11)
        End If
        ' any interval
        If Level = 4 Then
            Intervals(n) = GenerateRandomNumber(0, 23)
        End If
    Next
End Sub

End Class

```

Personal Test Class

```

Class PersonalTest
    Inherits Test
    Private CurrentStudentID As String

    Sub New()
        MyBase.New(0, True)
    End Sub

    Overrides Sub GenerateIntervals()
        ' generates 10 intervals based on the average performance this student has had
        on each interval: the ones performed the most poor on will be tested
        Dim SQLString As String
        Dim CorrectCount(23) As Integer
        Dim TimesAttempted(23) As Integer
        Dim AverageSuccess As New List(Of Decimal)
        CurrentStudentID = LogInForm.GetCurrentID
        ' loops for all 24 intervals
        For interval = 0 To 23
            ' makes call to database and finds the number of times this interval has
            been identified correctly by this student
            SQLString = "SELECT COUNT(Correct) FROM tblResult
                        INNER JOIN tblQuestion
                        ON tblResult.TestID = tblQuestion.TestID
                        AND tblResult.QuestionNumber = tblQuestion.QuestionNumber
                        WHERE StudentID = @StudentID
                        AND IntervalID = @IntervalID
                        AND Correct = 1"
            Dim CorrectCountDB As New DatabaseConnection(SQLString)
            CorrectCount(interval) = CorrectCountDB.SelectTwoIDs(CurrentStudentID,
interval)

            'makes call to database and finds the number of times this student has
            been tested on this interval
            SQLString = "SELECT COUNT(Correct) FROM tblResult
                        INNER JOIN tblQuestion
                        ON tblResult.TestID = tblQuestion.TestID
                        AND tblResult.QuestionNumber = tblQuestion.QuestionNumber
                        WHERE StudentID = @StudentID
                        AND IntervalID = @IntervalID"
            Dim TimesAttemptedDB As New DatabaseConnection(SQLString)
            TimesAttempted(interval) = TimesAttemptedDB.SelectTwoIDs(CurrentStudentID,
interval)

            ' creates an average score based on number of times answered correctly and
            number of times tested total (worst being 0, best being 1)
            If TimesAttempted(interval) = 0 Then
                ' if the interval has never been tested it is assigned success of 0
                AverageSuccess.Add(0)
            End If
        Next
    End Sub
End Class

```

```

        ' new item is added for each interval containing success value
    Else AverageSuccess.Add(CorrectCount(interval) / TimesAttempted(interval))
    End If
Next
' saves the index (which represents the interval ID) to the list
Dim AverageSuccessWithIndex = AverageSuccess.Select(Function(n, i) New With
{.Num = n, .Idx = i})
' sorts the average success in ascending order
Dim sortedSuccesses = AverageSuccessWithIndex.OrderBy(Function(nwi) nwi.Num)
' saves the indexes (interval IDs) of the 10 intervals with the lowest success
and saves them to the Intervals array used in GenerateStartingNoteAndDirection()
Intervals = sortedSuccesses.Take(10).Select(Function(x) x.Idx).ToArray()
' randomises the order of the intervals to avoid the intervals being in order
(therefore making them easier to randomly guess)
Dim randomIndex As Integer
Dim tempvalue1 As Integer
Dim tempvalue2 As Integer
For x As Integer = 0 To Intervals.Length - 1
    Randomize()
    randomIndex = CInt(Int((Intervals.Length) * Rnd()))
    tempvalue1 = Intervals(x)
    tempvalue2 = Intervals(randomIndex)
    Intervals(x) = tempvalue2
    Intervals(randomIndex) = tempvalue1
Next
End Sub

```

End Class

Take Test Class

Class TakeTest

```

    Private TestID As String
    Private StudentID As String
    Private thisTest As New List(Of Question)

    Sub New(ByVal IDTest As String, ByVal IDStudent As String)
        TestID = IDTest
        StudentID = IDStudent
        GetQuestionsFromDatabase()
    End Sub

    Private Sub GetQuestionsFromDatabase()
        Dim intervalIndex As Integer
        Dim StartingNote As Integer
        Dim Direction As String
        Dim SQLString As String
        ' loops through each question number and gets the intervalID, starting note
        and direction associated with that question and TestID
        For questionNumber = 1 To 10
            SQLString = "SELECT IntervalID FROM tblQuestion WHERE QuestionNumber = " &
questionNumber & " AND TestID = @ID"
            Dim IntervalIDDB As New DatabaseConnection(SQLString)
            intervalIndex = IntervalIDDB.SelectStringID(TestID)

            SQLString = "SELECT StartingNote FROM tblQuestion WHERE QuestionNumber = "
& questionNumber & " AND TestID = @ID"
            Dim StartingNoteDB As New DatabaseConnection(SQLString)
            StartingNote = StartingNoteDB.SelectStringID(TestID)

            SQLString = "SELECT Direction FROM tblQuestion WHERE QuestionNumber = " &
questionNumber & " AND TestID = @ID"
            Dim DirectionDB As New DatabaseConnection(SQLString)

```

```

        Direction = DirectionDB.SelectStringID(TestID)

        ' this information is added as a new item in the list of the questions in
this test
        thisTest.Add(New Question(TestID, intervalIndex, questionNumber,
StartingNote, Direction))
    Next
End Sub

Public Sub PlayInterval(ByVal QuestionNumber As Integer)
    Dim semitones As Integer
    ' intervalIndex, startingnote and direction are retrieved for the question
passed in, from the list created in GetQuestionsFromDatabase
    Dim intervalIndex As Integer = thisTest(QuestionNumber - 1).IntervalIndex
    Dim startingNote As Integer = thisTest(QuestionNumber - 1).StartingNote
    Dim Direction As String = thisTest(QuestionNumber - 1).Direction
    Dim SQLString As String
    SQLString = "SELECT Semitones FROM tblInterval WHERE IntervalID = @IntervalID"
    Dim SemitonesDB As New DatabaseConnection(SQLString)
    semitones = SemitonesDB.SelectIntegerID(intervalIndex)
    ' interval to be tested is played passing in starting note and finishing note
(which is based on direction and semitones)
    If Direction = "Asc" Then
        Interval.PlayInterval(startingNote, startingNote + semitones)
    Else
        Interval.PlayInterval(startingNote, startingNote - semitones)
    End If
End Sub

Public Function GetTestID()
    Return TestID
End Function

End Class

```

Database Connection Class

```

Class DatabaseConnection
    Dim MyCommand As MySqlCommand
    Dim MyConnection As MySqlConnection
    Dim MyReader As MySqlDataReader
    Dim DDLstr As String
    Dim SQLstr As String
    ' -----HARD CODED FOR TESTING-----
    -----
    Dim ConnStr As String =
"server=localhost;user=testuser;Database=neatesting;port=3306;password=testing123;"
    ' The connection string has been hard coded for testing purposes - however in the
actual implementation of the program it would be stored in and read from a file
    ' -----
    -----

    Sub New(ByVal sqlString As String)
        ' saves SQLString and opens database for every connection instantiated
        SQLstr = sqlString
        OpenDatabase()
    End Sub

    Private Sub OpenDatabase()
        Try
            ' opens connection to database
            MyConnection = New MySqlConnection(ConnStr)
            MyConnection.Open()
        Catch ex As Exception
            Console.WriteLine(ex.Message)
        End Try
    End Sub
End Class

```

```

    Catch
        MsgBox("Could not connect to the specified database.")
    End Try
End Sub

Private Function ExecuteCommand()
    ' reads the results of the query and returns them
    Dim result As String = ""
    Try
        MyReader = MyCommand.ExecuteReader()
        While MyReader.Read()
            result = (MyReader.GetString(0))
        End While
    Catch ex As Exception
        MsgBox(ex.Message)
    End Try
    MyReader.Close()
    MyConnection.Close()
    Return result
End Function

' database queries with one paramater: int / str / str (returns list)

Public Function SelectIntegerID(ByVal IntervalID As Integer)
    ' creates command for query that needs a variable
    MyCommand = New MySqlCommand(SQLstr, MyConnection)
    MyCommand.Parameters.AddWithValue("@IntervalID", IntervalID)
    ' returns the value returned from the query
    Return ExecuteCommand()
End Function

Public Function SelectStringID(ByVal ID As String)
    ' creates command for query that needs TestID as a variable
    MyCommand = New MySqlCommand(SQLstr, MyConnection)
    MyCommand.Parameters.AddWithValue("@ID", ID)
    ' returns the value returned from the query
    Return ExecuteCommand()
End Function

Public Function GetClassList(ByVal TeacherID As String)
    ' returns a list using TeacherID as variable
    Dim result As New List(Of String)
    MyCommand = New MySqlCommand(SQLstr, MyConnection)
    MyCommand.Parameters.AddWithValue("@ID", TeacherID)
    Try
        MyReader = MyCommand.ExecuteReader()
        While MyReader.Read()
            result.Add(MyReader.GetString(0))
        End While
    Catch ex As Exception
        MsgBox(ex.Message)
    End Try
    MyReader.Close()
    MyConnection.Close()
    Return result
End Function

' database queries with two parameters: str & int / str & str

Public Function SelectTwoIDs(ByVal StudentID As String, ByVal IntervalID As Integer)
    ' creates command for query that needs IntervalID and StudentID as variables

```

```

    MyCommand = New MySqlCommand(SQLstr, MyConnection)
    MyCommand.Parameters.AddWithValue("@IntervalID", IntervalID)
    MyCommand.Parameters.AddWithValue("@StudentID", StudentID)
    ' returns the value returned from the query
    Return ExecuteCommand()
End Function

Public Function SearchUser(ByVal username As String, ByVal password As String)
    ' select using a username and password as variables
    MyCommand = New MySqlCommand(SQLstr, MyConnection)
    MyCommand.Parameters.AddWithValue("@Username", username)
    MyCommand.Parameters.AddWithValue("@Password", password)
    Return ExecuteCommand()
End Function

' database query with three parameters: str & date & str

Public Function SelectThreeStrings(ByVal TestName As String, ByVal DueDate As
Date, ByVal ClassID As String)
    MyCommand = New MySqlCommand(SQLstr, MyConnection)
    MyCommand.Parameters.AddWithValue("@TestName", TestName)
    MyCommand.Parameters.AddWithValue("@ClassID", ClassID)
    MyCommand.Parameters.AddWithValue("@DueDate", DueDate)
    ' returns the value returned from the query
    Return ExecuteCommand()
End Function

' non queries (delete/update)

Public Sub DeleteFromStudent(ByVal StudentID As String)
    ' executes non query with StudentID as variable
    Try
        MyCommand = New MySqlCommand(SQLstr, MyConnection)
        MyCommand.Parameters.AddWithValue("@ID", StudentID)
        MyCommand.ExecuteNonQuery()
    Catch ex As Exception
        MsgBox(ex.Message)
    End Try
    MyConnection.Close()
End Sub

Public Sub UpdateStudent(ByVal ClassID As String, ByVal StudentID As String)
    ' executes non query with StudentID and ClassID as variables
    Try
        MyCommand = New MySqlCommand(SQLstr, MyConnection)
        MyCommand.Parameters.AddWithValue("@ClassID", ClassID)
        MyCommand.Parameters.AddWithValue("@StudentID", StudentID)
        MyCommand.ExecuteNonQuery()
    Catch ex As Exception
        MsgBox(ex.Message)
    End Try
    MyConnection.Close()
End Sub

' save all values to tables

Public Sub SaveQuestions(ByVal TestID As String, ByVal QuestionNumber As Integer,
ByVal IntervalID As Integer, ByVal StartingNote As Integer, ByVal Direction As String)
    ' creates command for query to save to the question table that requires all of
the question table data as variables
    Try
        MyCommand = New MySqlCommand(SQLstr, MyConnection)

```

```

        MyCommand.Parameters.AddWithValue("@TestID", TestID)
        MyCommand.Parameters.AddWithValue("@QuestionNumber", QuestionNumber)
        MyCommand.Parameters.AddWithValue("@IntervalID", IntervalID)
        MyCommand.Parameters.AddWithValue("@StartingNote", StartingNote)
        MyCommand.Parameters.AddWithValue("@Direction", Direction)
        MyCommand.ExecuteNonQuery()
    Catch ex As Exception
        MsgBox(ex.Message)
    End Try
    MyConnection.Close()
End Sub

Public Sub SaveAssignment(ByVal TestName As String, ByVal TestID As String, ByVal
ClassID As String, ByVal DueDate As Date)
    ' creates command for query to save to the test table that requires all the
    test table data as variables
    Try
        MyCommand = New MySqlCommand(SQLstr, MyConnection)
        MyCommand.Parameters.AddWithValue("@TestName", TestName)
        MyCommand.Parameters.AddWithValue("@TestID", TestID)
        MyCommand.Parameters.AddWithValue("@ClassID", ClassID)
        MyCommand.Parameters.AddWithValue("@DueDate", DueDate)
        MyCommand.ExecuteNonQuery()
    Catch ex As Exception
        MsgBox(ex.Message)
    End Try
    MyConnection.Close()
End Sub

Public Sub SaveUser(ByVal ID As String, ByVal FirstName As String, ByVal Surname
As String, ByVal Password As String)
    ' saves all information into a user account (student or teacher)
    Try
        MyCommand = New MySqlCommand(SQLstr, MyConnection)
        MyCommand.Parameters.AddWithValue("@ID", ID)
        MyCommand.Parameters.AddWithValue("@FirstName", FirstName)
        MyCommand.Parameters.AddWithValue("@Surname", Surname)
        MyCommand.Parameters.AddWithValue("@Password", Password)
        MyCommand.ExecuteNonQuery()
    Catch ex As Exception
        MsgBox(ex.Message)
    End Try
    MyConnection.Close()
End Sub

Public Sub SaveClass(ByVal ClassID As String, ByVal ClassName As String, ByVal
TeacherID As String)
    ' save to class table
    Try
        MyCommand = New MySqlCommand(SQLstr, MyConnection)
        MyCommand.Parameters.AddWithValue("@ClassID", ClassID)
        MyCommand.Parameters.AddWithValue("@ClassName", ClassName)
        MyCommand.Parameters.AddWithValue("@TeacherID", TeacherID)
        MyCommand.ExecuteNonQuery()
    Catch ex As Exception
        MsgBox(ex.Message)
    End Try
    MyConnection.Close()
End Sub

```

```
Public Sub SaveResults(ByVal TestID As String, ByVal StudentID As String, ByVal
QuestionNumber As Integer, ByVal Correct As Boolean)
    ' save to results table
    Try
        MyCommand = New MySqlCommand(SQLstr, MyConnection)
        MyCommand.Parameters.AddWithValue("@TestID", TestID)
        MyCommand.Parameters.AddWithValue("@StudentID", StudentID)
        MyCommand.Parameters.AddWithValue("@QuestionNumber", QuestionNumber)
        MyCommand.Parameters.AddWithValue("@Correct", Correct)
        MyCommand.ExecuteNonQuery()
    Catch ex As Exception
        MsgBox(ex.Message)
    End Try
    MyConnection.Close()
End Sub

Public Sub SaveSummary(ByVal StudentID As String, ByVal TotalQuestions As Integer,
ByVal TotalCorrect As Integer, ByVal AverageScore As Integer)
    ' save to summary table
    Try
        MyCommand = New MySqlCommand(SQLstr, MyConnection)
        MyCommand.Parameters.AddWithValue("@StudentID", StudentID)
        MyCommand.Parameters.AddWithValue("@TotalQuestions", TotalQuestions)
        MyCommand.Parameters.AddWithValue("@TotalCorrect", TotalCorrect)
        MyCommand.Parameters.AddWithValue("@AverageScore", AverageScore)
        MyCommand.ExecuteNonQuery()
    Catch ex As Exception
        MsgBox(ex.Message)
    End Try
    MyConnection.Close()
End Sub

Public Sub SaveFinishedTest(ByVal TestID As String, ByVal StudentID As String,
ByVal Percentage As Integer, ByVal DateDone As DateTime)
    ' save to finished test table
    Try
        MyCommand = New MySqlCommand(SQLstr, MyConnection)
        MyCommand.Parameters.AddWithValue("@TestID", TestID)
        MyCommand.Parameters.AddWithValue("@StudentID", StudentID)
        MyCommand.Parameters.AddWithValue("@PercentageScore", Percentage)
        MyCommand.Parameters.AddWithValue("@DateCompleted", DateDone)
        MyCommand.ExecuteNonQuery()
    Catch ex As Exception
        MsgBox(ex.Message)
    End Try
    MyConnection.Close()
End Sub

' output data from database to table

Public Sub ClassTable(ByVal TeacherID As String)
    Try
        ' binds SQLtable to datagrid
        MyCommand = New MySqlCommand(SQLstr, MyConnection)
        MyCommand.Parameters.AddWithValue("@ID", TeacherID)
        MyCommand.CommandType = CommandType.Text
        Using sda As New MySqlDataAdapter(MyCommand)
            Using dt As New DataTable()
                sda.Fill(dt)
                TeacherMenu.DataGridView1.DataSource = dt
            End Using
        End Using
    End Using
```

```
        Catch
            MsgBox("Could not add data to table.")
        End Try
    End Sub

Public Sub ResultsTable(ByVal TestID As String, ByVal StudentID As String)
    Try
        ' binds SQLtable to datagrid
        MyCommand = New MySqlCommand(SQLstr, MyConnection)
        MyCommand.Parameters.AddWithValue("@TestID", TestID)
        MyCommand.Parameters.AddWithValue("@StudentID", StudentID)
        MyCommand.CommandType = CommandType.Text
        Using sda As New MySqlDataAdapter(MyCommand)
            Using dt As New DataTable()
                sda.Fill(dt)
                DisplayTestResultsForm.DataGridView1.DataSource = dt
            End Using
        End Using
    Catch
        MsgBox("Could not add data to table.")
    End Try
End Sub

Public Sub AssignmentsTable(ByVal ClassID As String, ByVal StudentID As String)
    Try
        ' binds SQLtable to datagrid
        MyCommand = New MySqlCommand(SQLstr, MyConnection)
        MyCommand.Parameters.AddWithValue("@ClassID", ClassID)
        MyCommand.Parameters.AddWithValue("@StudentID", StudentID)
        MyCommand.CommandType = CommandType.Text
        Using sda As New MySqlDataAdapter(MyCommand)
            Using dt As New DataTable()
                sda.Fill(dt)
                StudentMenu.dgvCurrentAssignments.DataSource = dt
            End Using
        End Using
    Catch
        MsgBox("Could not add data to table.")
    End Try
End Sub

Public Sub CompleteAssignmentsTable(ByVal ClassID As String, ByVal StudentID As
String)
    Try
        ' binds SQLtable to datagrid
        MyCommand = New MySqlCommand(SQLstr, MyConnection)
        MyCommand.Parameters.AddWithValue("@ClassID", ClassID)
        MyCommand.Parameters.AddWithValue("@StudentID", StudentID)
        MyCommand.CommandType = CommandType.Text
        Using sda As New MySqlDataAdapter(MyCommand)
            Using dt As New DataTable()
                sda.Fill(dt)
                StudentMenu.dgvCompleteAssignments.DataSource = dt
            End Using
        End Using
    Catch
        MsgBox("Could not add data to table.")
    End Try
End Sub

Public Sub SummaryTable(ByVal StudentID As String)
    Try
```



```

        ' binds SQLtable to datagrid
        MyCommand = New MySqlCommand(SQLstr, MyConnection)
        MyCommand.Parameters.AddWithValue("@StudentID", StudentID)
        MyCommand.CommandType = CommandType.Text
        Using sda As New MySqlDataAdapter(MyCommand)
            Using dt As New DataTable()
                sda.Fill(dt)
                StudentMenu.dgvTestSummaries.DataSource = dt
            End Using
        End Using
    Catch
        MsgBox("Could not add data to table.")
    End Try
End Sub

Public Sub IntervalTable(ByVal StudentID As String)
    Try
        ' binds SQLtable to datagrid
        MyCommand = New MySqlCommand(SQLstr, MyConnection)
        MyCommand.Parameters.AddWithValue("@StudentID", StudentID)
        MyCommand.CommandType = CommandType.Text
        Using sda As New MySqlDataAdapter(MyCommand)
            Using dt As New DataTable()
                sda.Fill(dt)
                IntervalData.dgvIntervalData.DataSource = dt
            End Using
        End Using
    Catch
        MsgBox("Could not add data to table.")
    End Try
End Sub

Public Sub StudentTable(ByVal TeacherID As String)
    Try
        ' binds SQLtable to datagrid
        MyCommand = New MySqlCommand(SQLstr, MyConnection)
        MyCommand.Parameters.AddWithValue("@TeacherID", TeacherID)
        MyCommand.CommandType = CommandType.Text
        Using sda As New MySqlDataAdapter(MyCommand)
            Using dt As New DataTable()
                sda.Fill(dt)
                TeacherMenu.dgvStudentResults.DataSource = dt
            End Using
        End Using
    Catch
        MsgBox("Could not add data to table.")
    End Try
End Sub

' output data from database to chart

Public Sub SummaryChart(ByVal StudentID As String)
    Try
        ' adds data to the summary chart from the database
        StudentMenu.crtSummaryData.Series("% Score").Points.Clear()
        MyCommand = New MySqlCommand(SQLstr, MyConnection)
        MyCommand.Parameters.AddWithValue("@StudentID", StudentID)
        MyReader = MyCommand.ExecuteReader
        While MyReader.Read
            StudentMenu.crtSummaryData.Series("%
Score").Points.AddXY(MyReader.GetDateTime("DateCompleted"),
MyReader.GetInt32("PercentageScore"))

```

```

        End While
    Catch
        MsgBox("Could not add data to chart.")
    End Try
End Sub

Public Sub StudentChart(ByVal StudentID As String)
    ' adds data to the summary chart from the database
    Try
        StudentMenu.crtSummaryData.Series("% Score").Points.Clear()
        MyCommand = New MySqlCommand(SQLstr, MyConnection)
        MyCommand.Parameters.AddWithValue("@StudentID", StudentID)
        MyReader = MyCommand.ExecuteReader
        While MyReader.Read
            StudentGraph.crtStudentData.Series("%
Score").Points.AddXY(MyReader.GetDateTime("DateCompleted"),
MyReader.GetInt32("PercentageScore"))
        End While
    Catch
        MsgBox("Could not add data to chart.")
    End Try
End Sub

End Class

Interval Class
Class Interval

    Protected Shared Sub Play(interval() As Note)
        ' this plays the interval
        Dim n As Note
        For Each n In interval
            If n.NoteTone = Tone.REST Then
                Thread.Sleep(CInt(n.NoteDuration))
            Else
                Console.Beep(CInt(n.NoteTone), CInt(n.NoteDuration))
            End If
        Next n
    End Sub

    Public Shared Sub PlayInterval(ByRef StartingNote As Integer, ByRef EndingNote As
Integer)
        ' each note is in the array notes
        Dim Notes() As String = {Tone.C3, Tone.Cs3, Tone.D3, Tone.Ds3, Tone.E3,
Tone.F3, Tone.Fs3, Tone.G3, Tone.Gs3, Tone.A3, Tone.As3, Tone.B3, Tone.C4, Tone.Cs4,
Tone.D4, Tone.Ds4, Tone.E4, Tone.F4, Tone.Fs4, Tone.G4, Tone.Gs4, Tone.A4, Tone.As4,
Tone.B4, Tone.C5, Tone.Cs5, Tone.D5, Tone.Ds5, Tone.E5, Tone.F5, Tone.Fs5, Tone.G5,
Tone.Gs5, Tone.A5, Tone.As5, Tone.B5, Tone.C6}
        ' a new interval is created with the starting note and ending note
        Dim PlayInterval As Note() = {
            New Note(Notes(StartingNote), Duration.HALF), New Note(Notes(EndingNote),
Duration.HALF)}
        ' the interval is played
        Play(PlayInterval)
    End Sub

    Protected Enum Tone
        ' list of note names and their respective frequencies
        REST = 0
        C3 = 130.81
        Cs3 = 138.59
        D3 = 146.83

```

```

Ds3 = 155.56
E3 = 164.81
F3 = 174.61
Fs3 = 185
G3 = 196
Gs3 = 207.65
A3 = 220
As3 = 233.08
B3 = 246.94
C4 = 261.63
Cs4 = 277.18
D4 = 293.66
Ds4 = 311.13
E4 = 329.63
F4 = 349.23
Fs4 = 369.99
G4 = 392
Gs4 = 415.3
A4 = 440
As4 = 466.16
B4 = 493.88
C5 = 523.25
Cs5 = 554.37
D5 = 587.33
Ds5 = 622.25
E5 = 659.25
F5 = 698.46
Fs5 = 739.99
G5 = 783.99
Gs5 = 830.61
A5 = 880
As5 = 932.33
B5 = 987.77
C6 = 1046.5

```

```
End Enum
```

```
Protected Enum Duration
```

```
' list of note durations so that the length of the notes being played can be
changed
```

```

WHOLE = 1600
HALF = WHOLE / 2
QUARTER = HALF / 2
EIGHTH = QUARTER / 2
SIXTEENTH = EIGHTH / 2

```

```
End Enum
```

```
Protected Structure Note
```

```
' when a new note is created it is assigned a pitch and a duration
```

```
Private Tonevalue As Tone
```

```
Private Durationvalue As Duration
```

```
Public Sub New(frequency As Tone, time As Duration)
```

```
    Tonevalue = frequency
```

```
    Durationvalue = time
```

```
End Sub
```

```
Public ReadOnly Property NoteTone() As Tone
```

```
    Get
```

```
        Return Tonevalue
```

```
    End Get
```

```
End Property
```

```
        Public ReadOnly Property NoteDuration() As Duration
            Get
                Return Durationvalue
            End Get
        End Property
    End Structure
End Class
```