Design

**Design**

**Brief overall system description:**

My proposed system is a revision/teaching tool for GCSE maths. There are 3 different users, the administrator, teacher and student, each having different privileges which effect how they can use the system.

The role of the administrator would be to manage the user accounts in the system. The admin can add teacher, student and admin accounts and is able to create new classes in the system. For example, the admin can create a new class (e.g. Block C, Room 152), a new teacher (Alan Sugar) and assign that teacher to block C. This means that Alan Sugar is now a teacher who teaches in block C.

The admin could also create student accounts. For example the admin creates the student Joseph Evans, then assigns Joseph to block C, which means that Joseph will be in the class of all the teachers who teach block C. So Joseph will be in Alan Sugar’s class and all the other teachers assigned to teach in block B.

The main role of the teacher is to manage the learning of their students. One way they can do this is by adding a variety of GCSE Maths questions for students to answer, either manually using a form or automatically using a question generator feature. To generate questions automatically the teacher would fill in a criteria on a form which generates questions based on the criteria. The teacher can use this feature to generate a variety of questions of different topics and styles (e.g. multiple choice questions). For example, if the teacher selects “Algebra” as the chosen topic, “B” as the chosen grade difficulty, 12 as the chosen amount and multiple choice as the chosen question style, then the system will generate 12 multiple choice algebra questions of grade difficulty B and add it to the database automatically.

The teacher could also manage questions by being able to view, edit update or delete any of the existing questions in the database. In addition, the teacher could select a set of existing questions and add them to a quiz. Students can then use the system to complete all the quizzes set by their teacher. The teacher could then use a feature to search for the students assigned to their class and view their performance in all the quizzes they have completed.

The main role of the student is to practise questions. This is done by them completing any outstanding quizzes set by their teacher. Once a quiz is completed the result is returned to their teacher. If they have no outstanding quizzes to complete they can practise a random set of questions in system, but the result doesn’t get sent to the teacher.

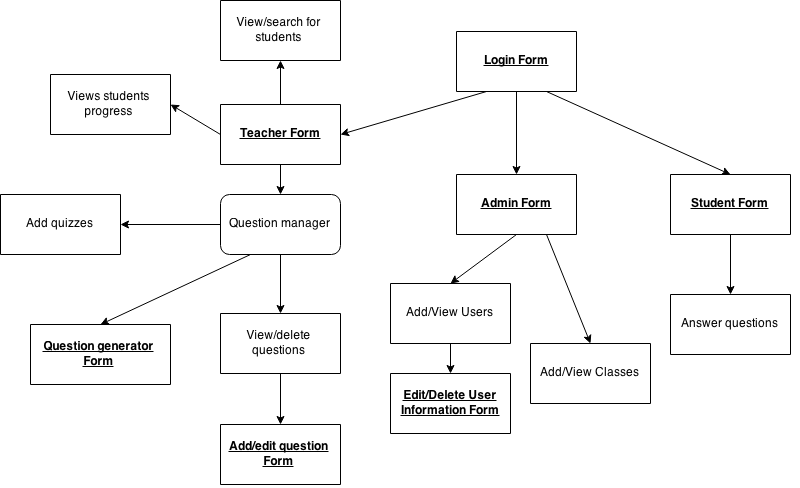
|  |  |  |  |
| --- | --- | --- | --- |
| **Inputs** | **Processes** | **Storage** | **Outputs** |
| **Add New User**  UserID  Firstname  Lastname  Rank | Uses an algorithm which generates a username and password based upon the first name and last name entered. | SQL User table | A message box showing whether the action performed was successful or not |
| **Add New Class**  Block Name  Room Number | Adds a new class and assigns a teacher to that class. | SQL User table  SQL Class table | A message box showing whether the action performed was successful or not |
| **Add New Question**  Question  Topic, Grade  Answer  Multiple Choice | Validates to check if some mandatory text boxes are not empty and if it’s not it will insert the question into database | SQL Question table | A message box showing whether the action performed was successful or not |
| **Add New Quiz**  Title  QuestionID | Adds a set of existing questions to a new quiz. | SQL Quiz Table | A message box will be displayed to indicate that the operation has been successful. |

Overall System Design - IOPS Chart

Modular Design - Form/Navigation

**Login Form**

1. **Admin Form**
   * + **Add/view User**
       - Edit/Delete User Information
     + **Add/view Classes**
2. **Teacher Form**
   * + **View students’ progress**
     + **View/search for students**
     + **Question Manager**
       - Add/edit question
       - View/delete questions
       - Generate questions
       - Add quizzes
     + **Generate questions**
3. **Student Form**
   * + **Answer questions**

This is a diagram of the overall navigation in my system

Data design dictionary

tblUser

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Field Purpose** | **Data Type** | **Field Size** | **Example Data** | **Validation** |
| UserID | A unique Identifier for the user | Integer | 10 | 30034876 | Must be an Integer and not be left blank |
| Firstname | Stores first name of the user | Varchar | 30 | Alan | Must not be blank |
| Lastname | Stores last name of the user | Varchar | 30 | Sugar | Must not be blank |
| Email | Stores user’s college email address | Varchar | 40 | asugar@sfx.ac.uk | Must not be blank and must be a valid email containing “@” |
| Picture | Stores the picture of each user | Image | 1MB |  | Must contain a valid image |
| Username | Stores the users unique username for the ”MyGCSE Maths” system | Varchar | 20 | elijaho7612 | Must not be blank |
| Password | Stores users password for the ”MyGCSE Maths” system | Varchar | 20 | Francis55Xav | Must not be blank |
| Rank | The Rank of the current User. e.g. Admin, Student or Teacher  0 - Student  1 - Teacher  2 - Admin | Integer | 3 | 1 | The rank cannot contain value <0 Or > 2 |

tblClass

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Field Purpose** | **Field Type** | **Field Size** | **Example Data** | **Validation** |
| ClassID | A unique Identifier for each class | Integer | 2 | 23 | Must be an Integer and not be left blank |
| Block | The block the class is in | Char | 1 | B | Cannot be empty & must be a char. |
| Room | Stores the room location of the class. | Varchar | 6 | 101A | Must not be left empty |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Field Purpose** | **Field Type** | **Field Size** | **Example Data** | **Validation** |
| UserID | Stores the Unique UserID of the user | Integer | 10 | 44876 | Value cannot be null and must be an Integer |
| ClassID | The Unique ID of the class the user is assigned to. | Integer | 4 | 23 | Value cannot be null and must be an Integer |

tblUserClass

tblQuestion

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Field Purpose** | **Field Type** | **Field Size** | **Example Data** | **Validation** |
| QID | The unique identifier for each question | AutoNum/Int | 4 | 42 | Value cannot be null and must be unique |
| Topic | Stores the Topic of the question. | Varchar | 20 | Algebra | Value cannot be null |
| Grade | Stores a char representing the grade difficulty of the question. | Char | 1 | B | Cannot contain an invalid char like “Z” or “Y” since those are not a valid grades |
| Question | This is the question. | Varchar | 100 | “What is the square root of 5?” | Value cannot be left blank |
| Image | Some questions are better answered when you can see an image of what is required. This field stores the image. | Image/Byte | 2MB |  | Data must contain a valid Image file |
| Answer | The answer to the question | Varchar | 50 | X=5 | Must not be blank |
| MultiChoice | Indicates if the question is multiple choice | Boolean | 1 | True | Must not be blank |

tblWrongAnswer

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Field Purpose** | **Field Type** | **Field Size** | **Example Data** | **Validation** |
| QID | The ID of the question the record is about | Integer | 4 | 42 | Value cannot be null and must be an Integer |
| Dummy1 | Used to store the false answers of a multiple choice question. | Varchar | 50 | 6x+23y | Must not be blank |
| Dummy2 | Used to store the false answers of a multiple choice question. | Varchar | 50 | 89x+13y | Must not be blank |
| Dummy3 | Used to store the false answers of a multiple choice question. | Varchar | 50 | x+23y | Must not be blank |

tblQuizTitle

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Field Purpose** | **Field Type** | **Field Size** | **Example Data** | **Validation** |
| QuizID | The Unique identifier of the quiz | Integer | 4 | 21 | Value cannot be null and must be an Integer |
| Title | Holds the name of the quiz given by the teacher | Varchar | 40 | Week 4 Test | Value cannot be null |

tblQuiz

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Field Purpose** | **Field Type** | **Field Size** | **Example Data** | **Validation** |
| QuizID | Stores the Unique Identifier for the Quiz | Integer | 4 | 2 | Value cannot be null and must be an Integer |
| QID | Stores the ID of the question that is part of the quiz | Integer | 4 | 23 | Value cannot be null and must be an Integer |

tblQuizLog

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Field Purpose** | **Field Type** | **Field Size** | **Example Data** | **Validation** |
| UserID | Stores the unique ID of the student the record is about. | Integer | 8 | 4432 | Value cannot be null |
| QuizID | The unique Identifier for the quiz they have completed | Integer | 4 | 42 | Must not be blank |
| Mark | Stores what mark they achieved | Integer | 10 | 6 | Must be an Integer |
| Total | Stores the total mark possible for the question | Integer | 10 | 10 | Must be an Integer |
| DateCompleted | Stores the date the quiz was completed | DateTime | 8 | 12/08/2014 | Must contain a valid date and cannot be null |

Data Volumes

Potentially, my system would be storing information on 260 students and 10 teachers’. It will also be storing hundreds or thousands of questions in the SQL database.

Key: String = 1 byte per character | Integer = 4 bytes | Boolean = 1 bit

tblUser:

UserID = 10, Firstname = 30, Lastname = 30, Email = 40, Picture = 1MB, Username= 20, Password = 20, Rank = 3

**Total = 4bytes + 30 \* 2 bytes + 20 \* 2bytes + 1MB + 40 + 8 = 1.01MB**

tblClass

ClassID = 2, Block = 1, Room = 6

**Total = 4bytes + 1byte + 6 bytes = 11 Bytes**

tblUserClass

UserID = 8, ClassID = 8 : **Total = 8 bytes**

tblQuestion

QID = 4, Topic = 20, Grade = 1, Question = 100, Answer = 50, Image = 2MB, MultiChoice = 1

**Total = 4bytes + 20 bytes + 1 byte + 100bytes +50 bytes + 2MB + 1bit = 2.1MB**

tblWrongAnswer

QID = 4, Dummy = 50 \* 3

**Total = 4bytes + 150 Bytes = 154 Bytes**

tblQuiz

QuizID = 4, QID = 4,

**Total = 4 bytes + 4 bytes = 8 Bytes**

tblQuizTitle

QuizID = 4, Title = 40

**Total = 4 bytes + 40 bytes = 44 Bytes**

tblQuizLog

UserID = 10, QuizID = 4, Mark= 10, Total = 10, DateCompleted = 8

**Total = 4bytes + 4 bytes + 4 bytes + 4 bytes + 8 bit = 24 Bytes**

If I add tblUser, tblClass, tblUserClass (the user data tables) the sum would be **1.10 MB** per user.

Currently there are 260 potential users 1.10MB \* 260 = **286MB.** So an estimate of 286MB of data would be used in storing user information.

If I add all the question related tables up. It equals to **2.13MB** per question. So an average of **2.13MB** would be used for each question added.

There is a large amount of questions that could be added. But realistically, there probably won’t be more than 400 questions in the database at once.

2.13 \* 400 potential questions = **856 MB**

So an estimate of **856 MB** of would be used for storing 400 potential questions.

Therefore a grand total of: 856MB + 286MB = **1.14GB** would be used for the data storage in the SQL database.

Definition of record structure

I will be using custom structures/objects throughout my coded solution. I am planning to have the structures user and question in my program. The attributes in these structures will be identical to the fields in the User and Question table in my SQL database. A scenario where I will make use of structures is when retrieving questions from SQL tables, I could store all the information about one question into a question object/structure.

The structure User would contain the following attributes.

Create Structure User

var UserID, ClassID

var Username, Password

var Firstname , Lastname

var Email

var Rank

var Image

The structure Question would contain the following members.

Create Structure Question

var ID

var QuestionString, Answer

var Topic, Grade

var isMultiChoice, falseAnswers()

Note: These custom objects have the same attributes as the tblQuestion and tblUser SQL tables.

I have adopted this approach of using custom datatypes and objects because it produces code that is easy to read, manipulate and maintain. This makes implementing complex features easier since the code is well written and well standardised.

In addition to structures, I will also be making use of text files (.txt). These will be used to store data to the end user’s computer (teacher) from the system when they want to export mark schemes/answers to the questions/quizzes stored in the system. An algorithm will be used to write the data to a text file in the correct format, as shown below.

1. Question: Multiply out the brackets and simply (b + 5)(b - 7)

Answer: b² -2b -35

I will also be using a text file (.txt) to save the data for when a teacher exports the MyGCSE Maths account credentials for each of the student they teach.

The teacher could then easily hand out login information to the students in class if they need to. The file would be structured in the format shown below.

1. Fullname: Ben Davis

Username: davisb4887

Password: dboy21

Normalized database

This is my initial table (not normalized).

Table1 (UserID, Firstname, Lastname, Email, Picture, Username, Password, Rank, ClassID, Block, Room, QID, Topic, Grade, Question, Answer Image, MultiChoice, Dummy1, Dummy2, Dummy 3, QuizID, Title, Mark, Total, DateCompleted)

First normal form (1NF)

**To reach first normal form we must split up or remove any table that has any repeating attributes.**

tblUser (UserID, Firstname, Lastname, Email, Picture, Username, Password, Rank, ClassID, Block, Room)

tblQuestion (QID, Topic, Grade, Question, Answer, Image, MultiChoice, Dummy1, Dummy2, Dummy3, QuizID, Title, Mark, Total, DateCompleted)

Second normal form (2NF)

**We continue to split up the tables so each table is unique**

tblUser (UserID, Firstname, Lastname, Email, Username, Password, Rank, ClassID)

tblClass (ClassID, Block, Room)

tblQuestion (QID, Topic, Grade, Question, Answer, Image, MultiChoice, Dummy1, Dumm2, Dummy3)

tblQuiz (QuizID, QID, Title, UserID, Mark, Total, DateCompleted)

Third normal form (3NF) \_- Fully Normalized

**It is now in third normal form because we have no many to many relationships, and all data is dependent on the primary key, and there are no repeating attributes or groups of data.**

tblUser (UserID, Firstname, Lastname, Email, Picture, Username, Password, Rank)

tbUserClass(UserID, ClassID)

tblClass (ClassID, Block, Room)

tblQuestion (QID, Topic, Grade, Question, Answer, Image, MultiChoice)

tblWrongAnswers (QID, Dummy1, Dummy2, Dummy3)

tblQuiz (QuizID, QID)

tblQuizTitle (QuizID, Title)

tblQuizLog (QuizID, UserID, Mark, Total, DateCompleted)

Normalized tables – Entity Relationship Diagram

tblUser (UserID, Firstname, Lastname, Email, Picture, Username, Password, Rank)

tblUserClass(UserID, ClassID)

tblClass (ClassID, Block, Room)

tblQuestion (QID, Topic, Grade, Question, Image, Answer, MultiChoice)

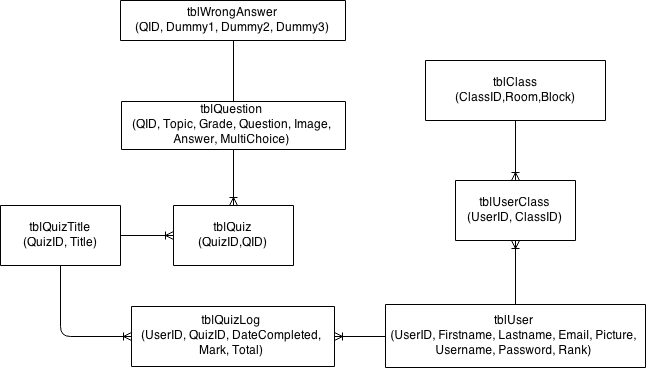
tblWrongAnswers (QID, Dummy1, Dummy2, Dummy3)

tblQuiz (QuizID, QID)

tblQuizTitle(QuizID, Title)

tblQuizLog (UserID, QuizID, Mark, Total, DateCompleted)

E-R Diagram

E-R Diagram

I inserted the table UserClass in-between User and Class remove the many to many relationship. I also inserted the table QuizLog in-between User and QuizTitle to remove the many to many relationship. Also, each question can appear in many quizzes and for each question, there can be one false answer record which contains multiple false answers.

Potential SQL Queries

|  |  |
| --- | --- |
| English description: | Selects all of the information of the user attempting to login. So the system will know if they’re authorised or not and if they are find relevant information about them. e.g. Firstname, Lastname, Rank etc. |
| Query: | SELECT \* FROM tblUser  WHERE Username =?  AND [Password] =? |

|  |  |
| --- | --- |
| English description: | Selects 10 random questions from the database. |
| Query: | Select Top 10 \* From tblQuestion Order By NEWID() |

|  |  |
| --- | --- |
| English description: | An SQL query that adds new users information into the users table. |
| Query: | INSERT INTO tblUser (UserID, Firstname, Lastname, Email, Picture, Username, Rank) VALUES (?,?,?,?,? ? ) |

|  |  |
| --- | --- |
| English description: | Inserts a question to the SQL table |
| Query: | INSERT INTO tblQuestion(Topic, Grade, Question, Image, Answer MultiChoice)  Values(?,?,?,?,?,?) |

|  |  |
| --- | --- |
| English description: | This updates a specific user’s information. |
| Query: | UPDATE tblUser  SET Firstname =?, Lastname = ?, Email = ?, Picture = ?  WHERE UserID = ? |

|  |  |
| --- | --- |
| English description: | Selects all the questions in the quizzes along with the title of the quiz from another table. |
| Query: | Select \* From tblQuiz, tblQuizTitle  Where tblQuiz.QuizID = tblQuizTitle.QuizID |

|  |  |
| --- | --- |
| English description: | Deletes a specific user from the user table |
| Query: | DELETE FROM tblUser WHERE UserID = ? |

|  |  |
| --- | --- |
| English description: | This is the query used when a teacher is searching for the students in their class by filtering students according to their name. This query then displays and orders the students found according to their first name in ascending order. |
| Query: | SELECT \* FROM tblUser, tblUserClass  WHERE tblUser.Firstname + ‘ ‘ + tblUser.Lastname LIKE '% ? %'  And tblUser.[Rank] = 0  And tblUser.UserID = tblUserClass.UserID  And tblUserClass.ClassID = ?  Order By tblUser.Firstname Asc |

|  |  |
| --- | --- |
| English description: | This query finds all the quizzes that a student has not yet completed. |
| Query: | SELECT Distinct tblQuiz.QuizID, tblQuizTitle.Title  FROM tblQuiz,tblQuizTitle  WHERE tblQuiz.QuizID = tblQuizTitle.QuizID  And tblQuiz.QuizID NOT IN (SELECT QuizID FROM tblQuizLog Where UserID = ?) |

Identification of storage & backup media

**Storage:**

I have calculated the size of the executable of my program to be around 600kb. In addition to the file size of the executable, the teachers would be exporting data from the SQL database to a .txt file on their computer. The teacher could export a “mark scheme” in the format of a .txt file which contains questions and answers for each question selected. This should not be any bigger than 1MB because the .txt files contain plain text which will not take up much space. It is unlikely that a teacher would export more than 50 questions per mark scheme at once, which is about 4kb in size.

With that in mind, there are many potential methods that could be used for storage. Most students and teachers own a USB and because my program does not need any file on the user’s computer to work, a storage system like a USB would be ideal as it allows users to store the executable of the file without it taking up much space. The disadvantage to using a USB is that my system will only work on the college’s local network, so it would be useless if they tried to use it at home.

Another solution is to store the executable of my program on Moodle in the GCSE maths section. The teacher could upload it to the shared maths resource folder, from where GCSE maths students could download the program onto their college PC account.

My chosen method of storage would be to store the program on colleges Moodle GCSE maths resource folder. This reduces the inconveniences of having to re-distribute the executable to each user manually and now the program can be available for students through one central location.

**Backup:**

I would back up the VB.NET project solution and the schema and data of the SQL database/tables. I will not need to back up the executable of my program because I could build/compile an executable of the solution at any given time from the VB.NET project solution files. I will back up my project on a USB stick and an online file-sharing website like dropbox. I could also backup the project files on another external computer system like a portable external HDD.

Algorithm Design

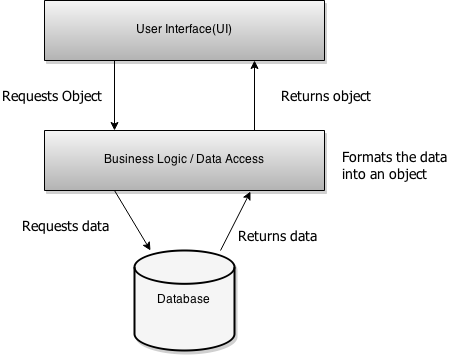
|  |  |  |
| --- | --- | --- |
| **Title** | **Description** | **Pseudo Code** |
| LoginCorrect | Checks if the credentials the user has entered into the Username textbox and Password textbox matches a record in the SQL database. If it is, it allows the user to login to the system. If the credential is not found then it says incorrect login. | If txtUsername And txtPassword IsNot Empty && Is In UserTable Then  LoginSuccessful()  ShowForm()  Else  Show error message “Your username or password is incorrect”  Finish If |
| ParseStr | A function that removes every non-alphabetic character in any given string using a regex pattern. This is used for the GenerateUsername and GenerateEmail functions. This is for keeping data integrity and for having data in the correct format. | Regex.Pattern = “[^a-zA-Z\- ]”  String final = Replace(Regex.Pattern, “ ”)  Return final |
| GenerateUsername | Generates the “MyGCSE Maths” account username based upon the rank, Firstname, Lastname and UserID that is passed down by the parameters Generally the algorithm works by extracting substrings. | If Rank.Student  Username = Lastname & Firstname.FirstLetter  Else  Username = Firstname.FirstLetter & Lastname  Finish If  Return Username |
| GeneratePassword | Generates a random password by running a For Loop 7 times and picking a random substring from the word variable and appending the substring it picks up to the password variable. At the end of the loop it returns the generated password. | String words = “abcdefghiklmnopqrstuwvxyz123456789”  String password  For Loop 7 Times  Int rnd = rand.Next(0, words.length)  Password += words.Substring(rnd,1)  End Loop  Return Password |
| GenerateEmail | Generates the email of a user. If the user is a student then it concatenates the first 7 letters of last name, first letter of first name and their user Id, otherwise it concatenates the first letter of Firstname followed by a full stop and the Lastname. It then returns the user email generated. | If Rank.Student  Email = Lastname & Firstname(0) & UserID & ”@student.sfx.ac.uk”  Else  Email = Firstname.FirstLetter & “.” & Lastname & “@sfx.ac.uk”  Finish If  Return Email |
| CreateQuizLbls  **Extra Info:**  This algorithm makes use of two custom objects I’ve created. Begin Structure Quizvar QuizIDvar TitleFinish StructureClass CustomLinkLabelInherit LinkLabelVar QuizIDFinish Class | This algorithm will be used for dynamically creating the link labels objects for quizzes that have not yet been completed by a student.  First, the algorithm calls a function that returns an array of data containing all the quizzes the student hasn’t yet completed from the SQL database. The data returned is of type Quiz, which is a custom object (see notation on the left).  The algorithm will then create a new CustomLinkLabel object for each item/quiz in the array using the data from the quizList array. After that the location and size of the link label is then defined. The Y Axis location of the label will be incremented upon each iteration, therefore the location of next label will be different to the previous one. Once 10 labels have been created, the algorithm will increment the X Location and reset the Y Location which places the label back to the top of the form.  At the end of each iteration an event handler for the click event will be added, which will allow students to start a quiz by clicking on the label.  The underlined labels below will be the end result of the algorithm. | List(Of Quiz) quizList = GetUnCompleteQuizzes(studentID)’A list is just a dynamic arrayint xLocation = 25int yLocation = 86int amountCreated = 0For i = Loop From 0 To quizList.count - 1Declare lblQuiz = New CustomLinkLabellblQuiz .QuizID = quizList(i).QuizIDlblQuiz .Text = quizList(i).TitlelblQuiz .Location = (xLocation, yLocation)lblQuiz .Size = (237, 32)amountCreated += 1yLocation += 33If amountCreated == 10 ThenxLocation += 350yLocation = 86amountCreated = 0Finish IfAddEvent lblQuiz.Click To StartQuiz()Controls.Add(lblQuiz) **End** Loop |
| CreateDynamicStudents  **Extra Info:**  This algorithm makes use of two objects I’ve created.  Begin Structure User  Var Firstname  Var Lastname  Var Username  Var UserID  Finish Structure Class CustomPictureBoxInherit PictureBoxVar UserIDVar UsernameVar PasswordFinish Class | This algorithm will be used to display picture boxes and labels in a defined pattern to the teacher. This algorithm will be used when the teacher wants to search for students in their class. The teacher enters the name they’re looking for into a text box and a function returns an array of the User object containing all the students found which satisfies the teachers search criteria.  The algorithm will loop through each item in the array and create a new CustomPictureBox control, setting the user Id, username and password of the picture box based upon the current item in the studentsList array. The X Location of each picture box is changed upon each iteration and once is 5 picture boxes have been created, the X location is reset and the Y location is incremented.  An event handler is then added so that when the teacher clicks the picture box, they can perform actions like viewing students’ performance.  The image below is an illustration of the end result of the use of this algorithm.  C:\Users\Toshiba\Desktop\Screenshot_2.pngEach picture box and label represents the student found that are assigned to the teachers class. | List(Of User) studentsList = GetStudents(classId, “name”)  int xLocation = 20  int yLocation = 20  int amountCreated = 0  int classId = Teacher.ClassID  For i = Loop From 0 To studentList.count  Declare picbox = New CustomPictureBox  picbox .UserID = studentsList(i).QuizID  picbox .Name = studentsList(i).Firstname & studentsList(i).Lastname  picbox .Location = (xLocation, yLocation)  picbox .Size = (167, 186)  picbox.Image = studentList(i).Image  amountCreated += 1  xLocation += 220  var lbl = New Label  lbl.Location = xLocation + 22 & yLocation += 190  AddControl(lbl)  If amountCreated == 5 Then  xLocation = 20  yLocation += 220  amountCreated = 0  Finish If  AddEvent picBox.Click -> ShowQuizResults()  AddControl(picBox) End Loop |
| RandomizeAnswer | This algorithm takes an array and randomizes the order of each item. The parameters passed is ByRef so the changes made will reflect on the original array. This will be used to randomize multiple choice answers to a question before displaying it to the user. | Begin Sub (ByRef ArrayA() As String)For i = 0 to ArrayA.Length - 1String temp = ArrayA(i)Int index = RandomInt(Array.Length – 1)ArrayA(i) = ArrayA(index)ArrayA(i) = tempFinish For LoopFinish Sub |
| GenerateShapesQuestion  **Extra Info:**  This algorithm makes use of three objects I’ve created.  Begin Structure TPoint  var Text  Int xLocation, yLocation  var Format  Finish Structure  Begin Enum shape  Square  Rectangle  Triangle  Finish Enum  Begin Enum QuestionType  Perimeter  Area  Finish Enum | This algorithm is used to generate random shape questions.  Three structures are defined inside the scope of the ShapeQuestion class. One of the structure is TPoint which represents the location where text will be displayed on a drawn image. The variable format indicates whether the text will be printed sideways, landscape or portrait.  The shapes structure is shape and question type(see notation on the left)  This algorithm can generate three types of question for square, rectangle and triangle shapes.  I will only show the triangle algorithm as it’s the most complex (see algorithm on the next page).  First the algorithm must choose what type of shape question to generate. This happens by the algorithm selecting a random shape from a shape array, generating a question according to the shape picked then formulating a string representation of the question.  Assuming the chosen shape was a triangle the algorithm on the next page describes the procedures that follows. | Global var Answer  Global Var FalseAnswers = Array()  Global var PicShape = PictureBox  Begin Sub GenerateQuestion()  var RandomShape = GetRandomShape  var QuesType = GetRandomQuestionType  If RandomShape == Shape.Rectangle  PicShape.Image = DrawTriangleQuestion  Else if RandomShape == Shape.Square  PicShape.Image = DrawSquareQuestion  Else if RandomShape == Shape.Triangle  PicShape.Image = DrawRectQuestion  Finish If  var words[] = “Work out the” , “Calculate the” “What is the”  var randomWord = words(R.Next(0, words.Length – 1)  String question = randomWord & “ “ QuesType “ of this “ & RandomShape.ToString()  Return question Finish Sub |
| DrawTriangleQuestion  **Extra Info:**  Begin Structure TPoint  var Text  Int xLocation, yLocation  StringFormatFlags Format  Finish Structure | This algorithm generates a triangle question. Three variables are declared, the hypotenuse, adjacent and opposite. These are the lengths of each side of the triangle.  A loop is used to generate random numbers for each of the side of the triangle. Mathematical laws state that the hypotenuse must be greater than the adjacent, which in turn is greater than the opposite. The loop will only exit if these conditions are fulfilled.  Three labels of TPoint are declared. The Text property holds the value of the triangle side along with the unit. The location of where the label should be drawn is then predefined.  An instance of the Bitmap object is created of the size (260, 260), this is called Canvas. This is where the algorithm will draw the triangle image on.  An array of 3 coordinates is then defined. Each coordinate defines the position of each of the 3 corners of the triangle.  The algorithm then uses the graphics classes in vb.net and the array of coordinates to draw the triangle onto the canvas. After the triangle is drawn, the labels lblHypotenouse, lblAdjacent and lblOpposite are then drawn onto the sides of the triangle on the canvas.  The location of the triangle and labels always stays the same. However it’s the values of the labels that always change. The value could be “4cm” or “91m^2”  After the algorithm has finished generating the image, it calls a method that calculates the answer to the question string. The next page shows the algorithm that calculates the answers. | var hypot, adjac, oppos  char unit = Random{“m” or “c”}  Do  Hypot = RandomNumFrom(5 To 20)  Adjac = RandomNumFrom(3 To 15)  Oppos = RandomNumFrom(2 To 10)  Do until hypot > adjac & oppos < adjac  var lblHypotenouse = {.Text = hypot & unit, .xLocation = 140, .yLocation = 100}  var lblAdjacent = {.Text = hypot & unit, .xLocation = 140, .yLocation = 100}  var lblOpposite = {.Text = hypot & unit, .xLocation = 140, .yLocation = 100}  var lblCollection = {lblHypotenouse, lblAdjacent, lblOpposite}  var canvas = size(260, 260)  var points(2,2) = {(58, 29), (58, 212), (214, 212)}  Graphics.draw(points)  For each label in lblCollection  Draw each label onto triangle  End For Each  Canvas.SaveImg()  Answer = SolveTriangleQuestion(question  type, hypot, adjac, opposite)  Return Canvas.Image() |
| SolveTriangleQuestion | This function is called by the DrawTriangleQuestion method and it takes in 5 arguments which it uses to calculate the answer to the generate question string.  It takes the questionType, all the sides of the triangle, and the unit of the triangle (e.g. “cm” or “m”).  It calculates the answers to the question, along with false answers for multiple questions support.  It assigns the false answers to an array then returns the correct answer. | var answer  If questiontype == area  answer = ((opposite \* adjacent) / 2) & unit & ”²”.  FalseAnwers(0) = hypo + opposite + adjacent) & unit  FalsweAnswers(1) = (opposite \* adjacent) & unit “²”  Else if questionType == perimeter  answer = ((opposite + adjacent + opposite ) / 2) & unit  FalsweAnswers(0) = ((opposite \* adjacent) / 2) & unit & ”²”.  FalseAnwers(1) = (hypo + opposite) & unit  finish if  FalseAnswers(2) = RandomNum(opposite, hypo) & unit Return answer |
| QuickSort  **Extra Info:**  Begin Structure StudentQuizInfo  var UserID  var Firstname,  var Lastname  var QuizScores[]  var Average | This quick sort algorithms sorts an array of the custom object StudentQuizInfo (see notation on the left). The algorithm sorts based upon the value of the ‘average’ variable.  Note: ‘arrayOfItem’ is an array of type StudentQuizInfo  This quick sort class has a method called Sort() which is public and thus can be accessed outside of the scope of the class. The sort method is then called when a list of items need to be sorted.  The algorithm first finds the initial pivot so it knows how to divide the items into sub lists which can then be sorted recursively.  The GetPivotPositon is where the moving of items takes place. The algorithm selects a pivot, and sorts each item on the left and right of the pivot.  After it has finished moving items, it then returns the index of the new pivot back to the SortList method. The SortList method then makes a recursive call that sorts each item in the sub list of the new pivot index returned.  The laws of computer science state that a recursion needs a conditional statement in order to exit the recursive loop. Therefore the recursive loop only stops when all the items in the array have been sorted. | Class QuickSort  Begin Sub Sort(arrayOfItems)  if Length(arrayOfItems) <= 1  return arrayOfItems  else  var sortedList = SortList(0, Length(arrayOfItems), ArrayOfItems)  Return sortedList  Finish if  Finish Sub  Begin Function SortList(int lo, int hi, arrayOfItems) Returns ->SortedList  If hi > lo then  Int initialPivot = GetPivotPosition(lo, hi, arrayOfItems)    SortList(lo, initialPivot - 1, array)  SortList(initialPivot + 1, hi, array)  Finish If  Return arrayOfItems  Finish Function  Function GetPivotPosition(int lo, int hi, arrayOfItems)  Int x = lo  Int y = hi - 1  Int pivot = arrayOfItems(hi).Avergae  Start Do  While arrayOfItems(i).Average <= Pivot And i < hi  i = i + 1  Finish While  While arrayOfItems(i).Average <= Pivot And j > lo  j = j + 1  Finish While  If array(i).Average > pivot then  Swap(I, j, arrayOfItems)  End If  Finish Do  Return i  Finish Function  Begin Sub Swap(I, j , ByRef arrayOfItems)  Var selecteditem = arrayOfItems(i)  ArrayOfItems(i) = arrayOfItems(j)  ArrayOfItems(j) = selectedStudent  Finish Sub  Finish Class |
| ToStream  - This extension function will be used for readability and simplicity sake. It’s merely an implementation that exercises preference. | This method is an extension function to the bytes array datatype. This function allows me to call ToStream on any variable that is an instance of the byte array data type.  I will use this when converting a byte array which contains raw image data into a memory stream/image. I can then present the image to the user visually.  I could use the function like  Dim img[ ] As Byte = GetUserImg  PicBox1.Image = Image.FromStream(img.ToStream())  This is a more natural way of calling the function than  PicBox1.Image = Image.FromSteam(ToStream(img)) | <Extension()>  Begin Function ToSteam(bytes[] data) this->Function Returns MemoryStream  If data == Is Empty Then  Return MemoryStream(DefaultImg)  Else  Return MemoryStream new (data)  Finish if  Finish Function |
| ToBytes  - This extension function will be used for readability and simplicity sake. It’s merely an implementation that exercises preference. | This method is an extension function to the Image object/data type. This function allows me to call ToStream on any variable that is an instance of the Image object/data type.  I will use this when converting an Image to an array of bytes which can then be stored in the SQL database.  This function is useful because it will return a default image if the image its converting is empty, therefore is a more efficient than using a TryCatch in preventing errors  The function can be called like imgData.ToBytes() | <Extension()>  Begin Function ToBytes (Image img) this->Function Returns Bytes[]  var MemStream = New Memory Stream  If img == Is Empty Then  DefaultIMG. SaveRaw to MemStream  Else  Img.SaveRaw To MemStream  Finish if  Return MemStream.GetBytes()  Finish Function |
| IsValidEmail | Validates the email address by parsing it using the Regular expression class in VB.NET. Returns true if the email is valid or false if it is not valid.  This function is used when the admin wants to edit the email of a user. The email is first validated and if it’s valid then it will process it further. | var email = “Emai here”  var emailPattern =  “[\w\d-+\_.]+@[\w.-]+\.[\w]{2,5}”  Boolean isMatch = Regex.Match(email, emailPattern)  Return isMatch |

### Class Definitions

|  |
| --- |
| Class Names |
| EntityBase |
| Admin Service  Teacher Service  Student Service  Login Service  QuestionHelper  QuickSort  Question Generator  {  Shapes Question,  Algebra Question,  Numbers Question, Handling Data Question  } |

Each user (admin, teacher, system) and major task will have their own class that will carry out a set of specific tasks, using the methods in that class to do user specific things. For example, the student service class will have methods like GetUncompletedQuizzes(), SaveQuizResult(), whereas the admin service class will have methods like AddUser(), GetUser(), DeleteUser() and all these methods will be called from the corresponding UI form.

The idea behind my class setup is that there are methods which assists in completing tasks in the UI layer.

For example, I have a class called QuestionHelper, which will be used for adding, editing, deleting or getting a question. Therefore I will have the methods AddQuestion(), DeleteQuestion() and GetQuestion(). These methods can be called from the UI to complete a task. The GetQuestion() method can be called with the parameter value of 2453 and the function will query the SQL database to return the specific question requested, the data returned is then formatted into the Question object then returned to the caller.

I have adopted this approach because it makes my code structured and well organized. The class setup I have described is the 3-Tier Application model. The UI layer will make calls to the methods in the business logic, then the business logic will query the database directly and return the response object back to the UI to be processed or manipulated. This means that the UI is not involved in the backend work, instead it calls a stored procedure.

**The image on the right illustrates the described 3 Tier model that will be implemented.**

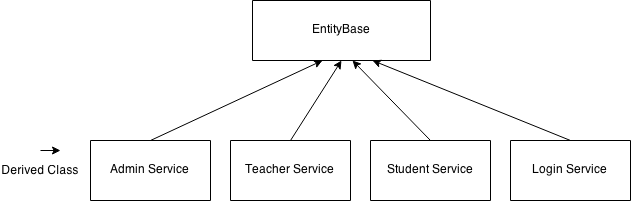
There will also be a class for generating questions which handles all the algorithms, methods, attributes and functions required for generating questions. When called, the class will generate a list of questions, then it calls the QuestionHelper class which adds all the generated questions to the SQL database.

Brief class outline

|  |  |  |
| --- | --- | --- |
| **Class Name** | **Description** | **Methods and nested classes** |
| EntityBase | The entity base class is where most classes will inherit from and it’s also where all the shared custom objects and methods will be declared.  Therefore, every derived class can make a reference to the objects and methods in the base class in order to use them. For example, in the Admin Service class, there will be a function called GetUsers() which returns an array of the user object which was originally defined in the EntityBase class.  The base class will also contain the SQLCmd and SQLConnection string which will be used each time any of the derived classes need to make an SQL connection. This is so the other classes don’t have to re-define the same connection string in their class, they can access the one in the Base class.  In the base class there will be objects that are shared throughout all derived classes.  For example, the variable MyUser is shared/static because it is needed to store the information of the user that is currently logged into system. | Public Object SQLCmd  Public Object SQLCon  Public String SQLConString  Public Sub: OpenConnection()  Public Sub: CloseConnection()  Public Extension Function: ToStream()  Public Extension Function: ToBytes()  Create Structure User  var UserID, ClassID  var Username, Password  var Firstname, Lastname  var Email  Rank UserRank  var Image  Create Enum Rank  Student = 0  Teacher = 1  Admin = 2  Global Shared MyUser As User  The MyUser variable represent the user that is currently logged in the system. |
| Admin Service | The admin service class is mainly used for managing user accounts in the system. It’s used to add, delete, and update user accounts in the SQL database. | Public Function: AddAccount()  Public Function: DeleteAccount()  Public Functuion: GetUsers()  Private Function: GenerateUsername()  Private Function: GeneratePassword()  Private Function: GenerateEmail() |
| Teacher Service | This class is used to perform teacher specific tasks. For example, CreateQuiz, GetUserResults and GetStudents | Public Function: GetStudentsResults()  Public Sub: ExportResults()  Public Function: GetStudents()  Public Sub: AddQuiz  Private Class : QuickSort |
| Student Service | The student service class is for performing student specific tasks. For example, a function for getting all the quizzes the student hasn’t completed. | Public Sub: Logout()  Public Function: GetUncompleteQuizzes()  Public Sub: SaveQuizResults() |
| Login Service | This class will manage users’ access to the system. For example, a function to check if the username and password entered at the login page is correct. If it is, allow the user to access the system. | Public Function: LoginCorrect()  Private Function: GetClassID()  Private Function: GetUserData() |
| QuestionHelper | This class will be used to manage questions. It is called when you need to add, edit, delete or to request a question. | Public Function: AddQuestion()  Public Function: EditQuestion()  Public Function: DeleteQuestion()  Public Function GetQuestion()  Private Function GetFalseAnswers() |
| QuestionGenerator | This class is used for generating questions. It has 4 nested classes which all generate a specific type of question. The list of generated questions is then returned to the caller. | Public Function: GenerateQuestion()  Private Class: AlgebraQuestion  Private Class: ShapesQuestion  Private Class: NumbersQuestion  Private Class: HandlingDataQuestions |

Inheritance

This diagram below illustrates the inheritance in my system.



The admin service, teacher service, student service, and login service will inherit all structures, objects, variables, methods from EntityBase which is the base class.

Inheritance allows for objects, methods and attributes to be shared by from the base class to the derived classes. This avoids re-declaring them within each class as this could lead to clustered code and ambiguity within the compiler.

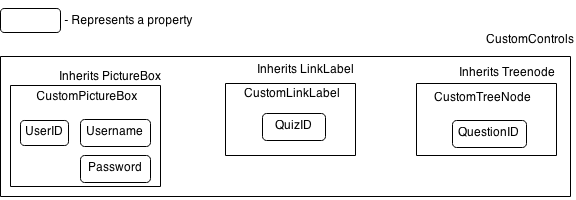
More Inheritance and user defined custom controls

I will have another class called CustomControls

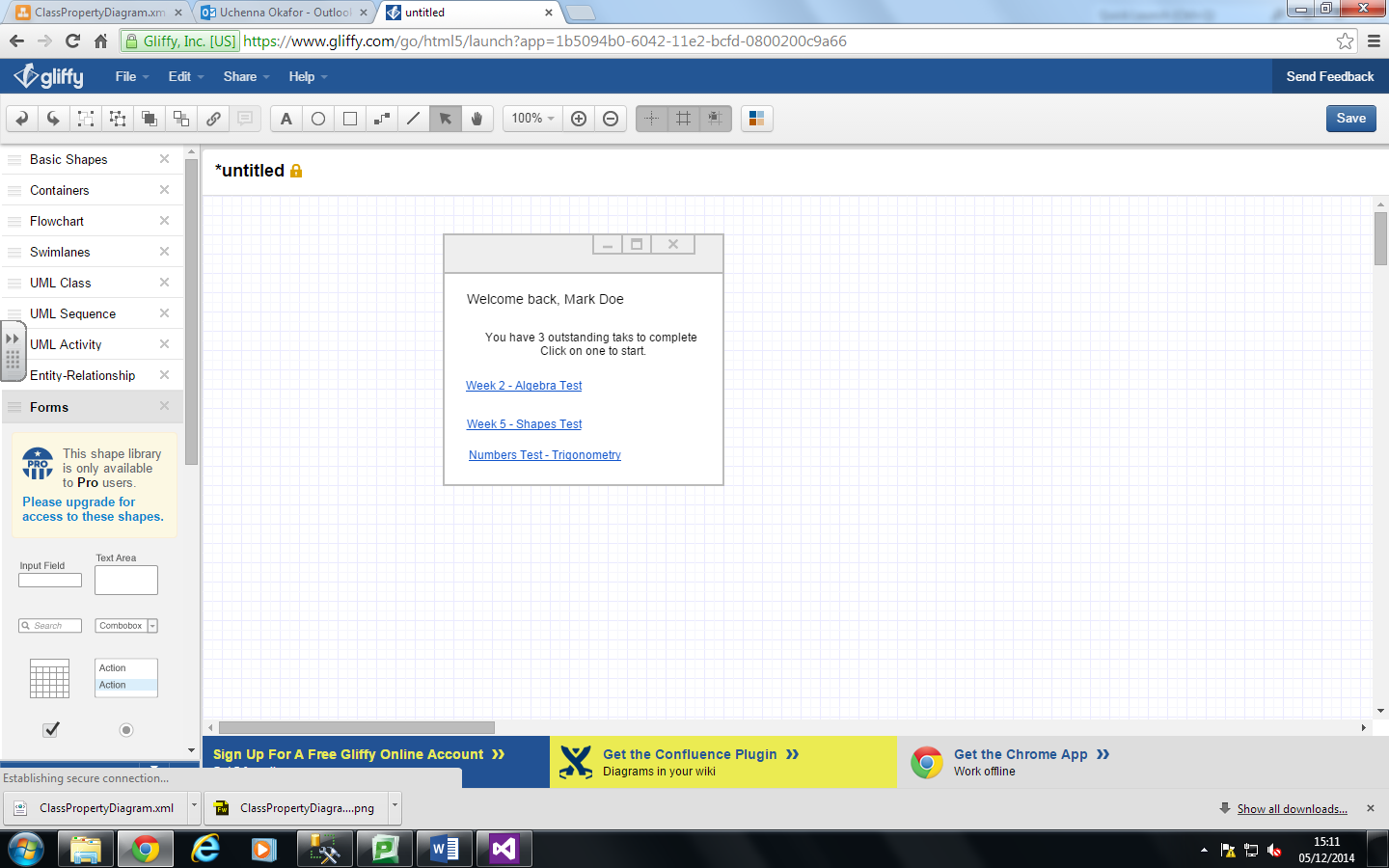
The CustomControls class contains 3 public nested classes which will each inherit former windows form UI controls such as picture boxes, tree nodes and link labels. I add my own properties and attributes to the classes which extends the features available. This takes full advantage of object orientation programming, such that when I Instantiate the custom classes, am able to use it as a more natural way of storing custom data on already existing UI controls like the picture box.

Inside the CustomControls class (Look at the diagram below), there will be a public class called CustomLinkLabel which will inherit the LinkLabel control. I will then add my own property called QuizID, used to store specific information which will make later stages of my programming easier. I will also have another Public Class in CustomControls called CustomTreeNode and CustomPictureBox, which will also have custom attributes used to help accomplish complex task when programming.

The diagram below illustrates the CustomControls class.

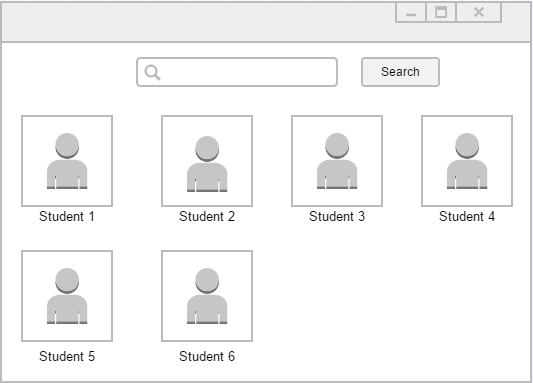


An example of where this could be useful is in the students form. I will instantiate the CustomLinkLabel object and use an algorithm to position the link labels on the screen at runtime. Each label will represent a quiz in the system that has not yet been completed by the student.

When a student has an outstanding task/quiz set by the teacher to complete, it will show up on their screen. This is where I can take advantage of the custom properties I’ve created. For each uncompleted quiz found in SQL quiz table, I can assign title of the quiz to the Text property of the CustomLinkLabel and the QuizID of the quiz to the .QuizID property of the CustomLinkLabel. Therefore, when the label is clicked the program will know the QuizID of the label without having to do any more unnecessary operations.

This shows each quiz that the student has not yet completed. Each label has a .QuizID property which stores the QuizID of the quiz. When the label is clicked, the program can easily access the .QuizID property directly from the label and perform a query to retrieve all the associated questions.

The use of custom UI controls would also be useful when the teacher wants to search for students in their class. The student’s data will be returned from the SQL database and each user found will be represented as a CustomPictureBox control. For example, I can assign the UserID of a student in the SQL database to the .UserID property of an instance of the CustomPictureBox control. Therefore, when the picture box is clicked, I could access the UserID and find out which student the picture box represents. I could then query the database to find out more information about that UserID. For example, the teacher could click a student’s picturebox and request for the quiz performances for that student. I will implement an algorithm to place these picture boxes in a pre-defined sequence (see CreateDyanamicStudents algorithm in design section).



Each picture box and label represents a student.

The label is the full name of the student, and the picture box is for storing the image of the pupil. Since each picture box is of the CustomPictureBox object, I can find the UserID of the student the picture box represents easily. I can then use the UserID to perform other tasks.

Use of polymorphism

In the BaseClass I will have a method called PromptLogout() which is called to logout a user from the system. It will prompt the user with a message saying “Are you sure you want to logout?” and if they agree it will log them out. I will override this in the Student Service class, so the message will say “Are you sure you want to logout? Any unfinished progress will be lost”. The student could be in the process of completing a quiz when they decide to logout and this will let them know that their progress is not saved if they haven’t finished.

Another place where I will be performing an override is in the custom objects I’ve created in the base class. I will be adding some of these objects to windows UI controls, and I’d need to override the ToString() function in order for it to show something meaningful instead of it showing the default object name. For example, I will be adding the User object to a ListBox control, so I will need to override the ToString() function to return the firstname and lastname of the object instead of the default object name. Which is MyGCSE\_Maths.EntityBase.User

Create Structure User

var UserID, ClassID

var Username, Password

var Firstname, Lastname

var Email

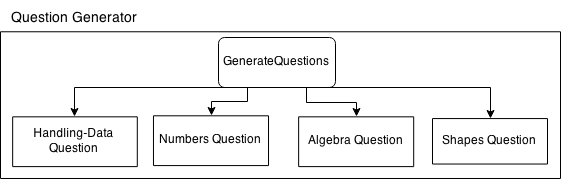
Rank UserRank

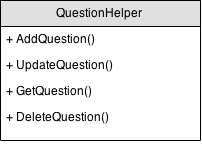
var Image

Public Overrides ToString()

Return Firstname & “ “ & Lastname

Other classes

Inside the Question Generator Class, there will be 4 private nested classes: Shapes Question, Handling Data Question, Numbers Question and Algebra Question. Each nested class can only be called by the GenerateQuestions method in the Question Generator Class. The 4 nested classes would all have a common function called GenerateQuestion() which when called, will generate a question and then return it to the GenerateQuestion method. This then will add all the generated questions from each nested class to an array of generated questions and return the array of generated question objects to the caller. The diagram below illustrates the class hierarchy for the Question Generator class.

The QuestionHelper class will be used to manipulate the questions that are stored in the SQL database. The QuestionHelper class has methods to add, get, delete or update a question. Each methods accepts arguments which are used in completing tasks/instructions. For example, the GetQuestion() method accepts 1 argument which is the ID of the requested question. When called, the GetQuestion() method will access the SQL database to get information of the question and return it in the form of the “Question” object. The DeleteQuestion() method accepts 1 argument which is the QuestionID. When called, it will delete the question associated with the requested ID from the SQL database.

### Use of abstract data structures:

### Queues:

### I will be making use of dynamic abstract data structures like Queues in the implementation of my system. I will in-cooperate a Queue for storing QuestionIDs of the questions that are yet to be answered by a student during a Quiz. For example, when students are taking part in a quiz, the questionIds associated with the quiz is loaded into a Queue. See diagram on the right.

### C:\Users\Toshiba\Downloads\Untitled Diagram.png

Next Question for student to answer.

Last Question to be answered.

### Each time the student answers a question, the Queue will be de-queued in order to retrieve the next QuestionID. The GetQuestion() method from the QuestionHelper class can then be called in order to retrieve the question associated with the QuestionID that had just been de-queued.

### When the Queue is empty, then the system will know that the students has finished all the questions in that quiz and will therefore show the student their score, and save it in the SQL database.

User Interface Rationale

I will be using a Graphical User Interface for simplicity and ease of navigation throughout my system. The two main factors I will be considering when designing my user interface is user friendliness and the users level of skill. ICT skills between users may vary greatly, so for that reason I shall place each object/control on my form in obvious places and I shall use images on buttons where appropriate in order to aid a user friendly interface.

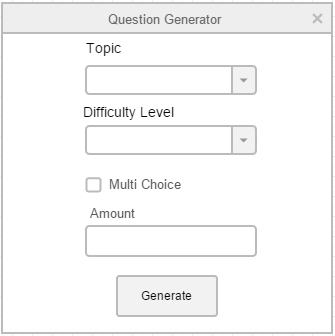
To make the system easy to navigate, use and robust, I shall make sure errors are correctly handled where appropriate and use sensible UI controls. I will use radio buttons, drop down lists in my program, this is so that it’s easy for the user to use the functionalities of the program. For example, having a drop down list would be an easier way to select an item as opposed to writing it out each time on a textbox/text field.

In addition I will keep the colour scheme of the system consistent throughout each form in the program. I will be using the default windows form colour scheme. The reason is because it’s very easy on the eyes, the colour scheme is nice, simple and elegant. Also, most windows users would already be already familiar with the default windows theme.

Finally, font size would vary throughout my system, but in general it would be appropriate relevant to the context, I will make sure the labels are not too small for users to see, and not too big that they would take up the majority of the screen.

User Interface Design

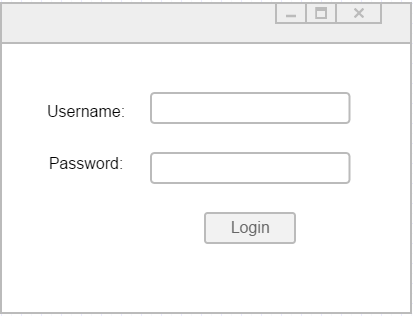
Teacher Form – Question Generator



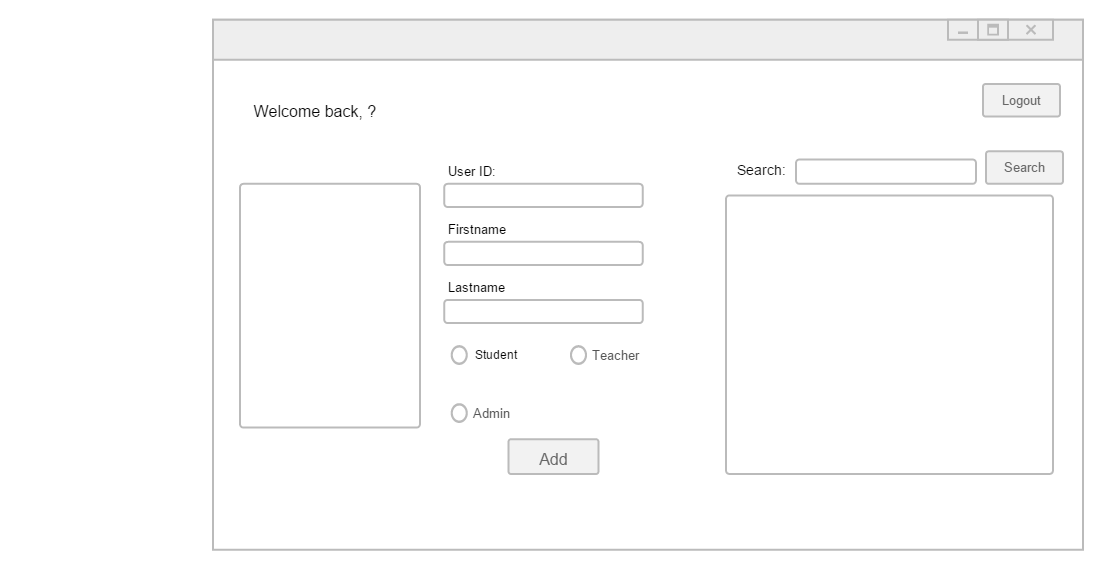
This form can only be accessed by teachers and is used to generate questions. They fill in the criteria, and click the generate button which will generate questions according to the criteria. I have used combo boxes to keep integrity of data, this is so the user cannot type in a topic that’s not valid. They can only select the items that are pre-defined/hard coded.

Login Form

When clicked, the button will use the username and password entered to attempt to log the user in.



This form will be used to add new users to the system. The admin enters the details of the user and clicks the add button which then adds the user to the database.

Admin Form – Add New User – This is a tabbed page consisting of 2 tabs

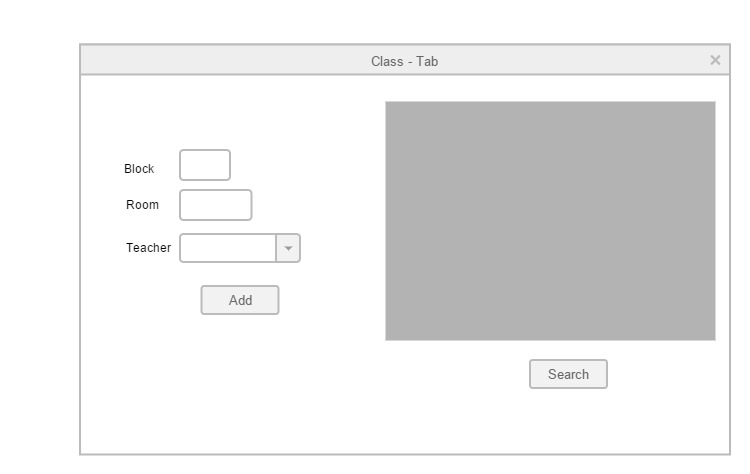
The list box will show all the users returned from the database according to the search criteria of the search textbox, once the search button is clicked it will perform the action. If a name in the list box is clicked it will open “frm Edit” on the next page with all of the users information displayed.

These radio buttons is to make it quick to assign a rank to a user and to prevent users from adding any invalid ranks that is not recognized.

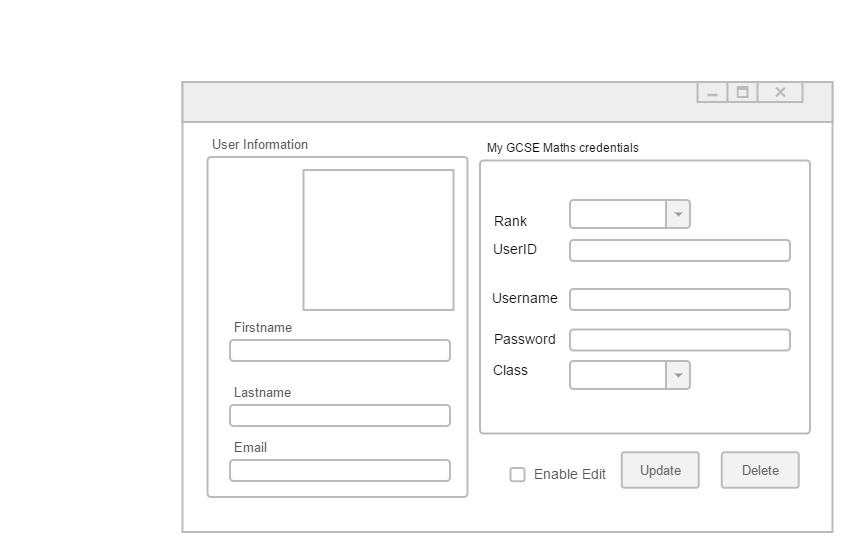
This is a Picture box frame, when clicked an OpenFileDialog will show allowing a picture to be selected and uploaded.

The block textbox is where the admin enters the block character of the room they want to add. The room text box is for entering what room number the class is located in. When the add button is clicked, the class will be created, meaning teachers and students can be assigned to those classes.

This is a data grid view. When the search button is clicked it will show all the classes that have been added to the system. It will also show any teachers that have been assigned to that class.

Admin Form – Add/View Classes – This is the second tab of the admin form

Admin form - frm Edit – Here admins can edit individual users

****

The rest of the text boxes, combo boxes, and picture box shows the relevant information of the user that has been selected.

When the CheckBox is checked it will allow the page to be edited thereby preventing data from being edited by accident.

Security & Integrity

**System Security**

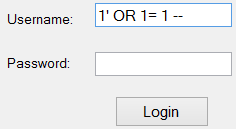
While I was implementing the login system for my new system I happened to stumble across a chapter in AQA Comp3 Wiki books about SQL Injection.

After reading it I tested the SQL Injection example from wiki books against a prototype of my system and discovered my system, especially the login panel was vulnerable to SQL Injection. Using the techniques of SQL injection I was able to login to my system without having or knowing credentials to the system, this also meant I could tamper with the database e.g. drop tables and edit tables without permission. This is an issue because students would be using my system and a potential student could attempt to gain unauthorized access to the system/database.

Initially my SQL login string was.

Select \* From tblUser Where Username = ‘?’And Password = ‘?’

An SQL injection attempt would look like 1' OR 1 = 1 --



This would cause the SQL login string to look like

Select \* From tblUser Where Username = '1' OR 1 = 1 -- And Password = ‘ ’

When executed against my system it would work because 1 = 1 is always going to return true and therefore give access to the user. If an unauthorized user could gain access to my system like this then it leaves the door wide open for malicious attacks.

I researched more and found out more on how SQL Injections worked and how to prevent them. I found that in order to prevent SQL injection attacks I should use parameterize/prepared SQL statements/queries. This was the security measure I took in order to secure my system.

After applying what I had learnt, my login SQL string looked like the following

Select \* From tblUser Where Username = @0 And Password = @1

SQLCmd.Paramaters.AddWithValue(“@0”, Username)

SQLCmd.Paramaters.AddWithValue(“@1”, Password)

After using parameterized SQL queries my system was more secure and no longer vulnerable to malicious SQL injection attacks. This means my system is safer and less likely for users to tamper with the data in the system. As a result of being made aware of potential vulnerabilities in my system, I shall parameterize every SQL statement that is executed in my system in order to prevent vulnerabilities.

In addition to SQL Injection prevention, another security measure I will be taking is to perform backups of the whole database every 2 weeks. A backup could also be made when there has been a new major update to the system. For example, I add an extra SQL table or I add a new feature to the system. This is so that if data gets corrupted or lost, there would be a backup that the database could be restored to.

Finally, I will implement features to prevent data from being modified by accident throughout my system. In order to do this, when an action is performed the system will ask the user if they’re sure they want to confirm the action before allowing them to continue.

**Integrity**

Throughout my system, I will take certain measures to prevent data captured from being corrupted.

One method I use is by using a Radio button and Combo Boxes with hard coded items. This way the end user cannot change them and add their own value. I will use this in the admin panel where new user accounts are created. The admin would only be able to assign the ranks admin, teacher and student. This way they cannot add any other rank other than the ones I have hard coded therefore keeping integrity of data and preventing the database from being corrupted with erroneous data

In addition to range validation, when entering a new user’s user Id, I will ensure that the user Id entered is an integers and if a non-integer is entered it will reject it and display an error message. Also, when entering Firstname, and Lastname I will use a Regular Expression (Regex) to remove every non alphabetic character found in their name, apart from “-“. This way if the admin was to add a new user and included an ampersand, comma, semi colon by accident it would automatically remove it, thus keeping integrity of data. Some people have double barrelled names, so therefore I will create the Regex to allow “-“. For example, if the first name entered is “Tho&^ma$%\*&s”, the regex pattern would return “Thomas”. Therefore keeping the integrity of the data.

My final course of action to keep data integrity is to provide a dialog box that will ask to user to confirm if they want to carry out the action they requested. If they choose yes the action will be performed. This prevents mistakes from being made.

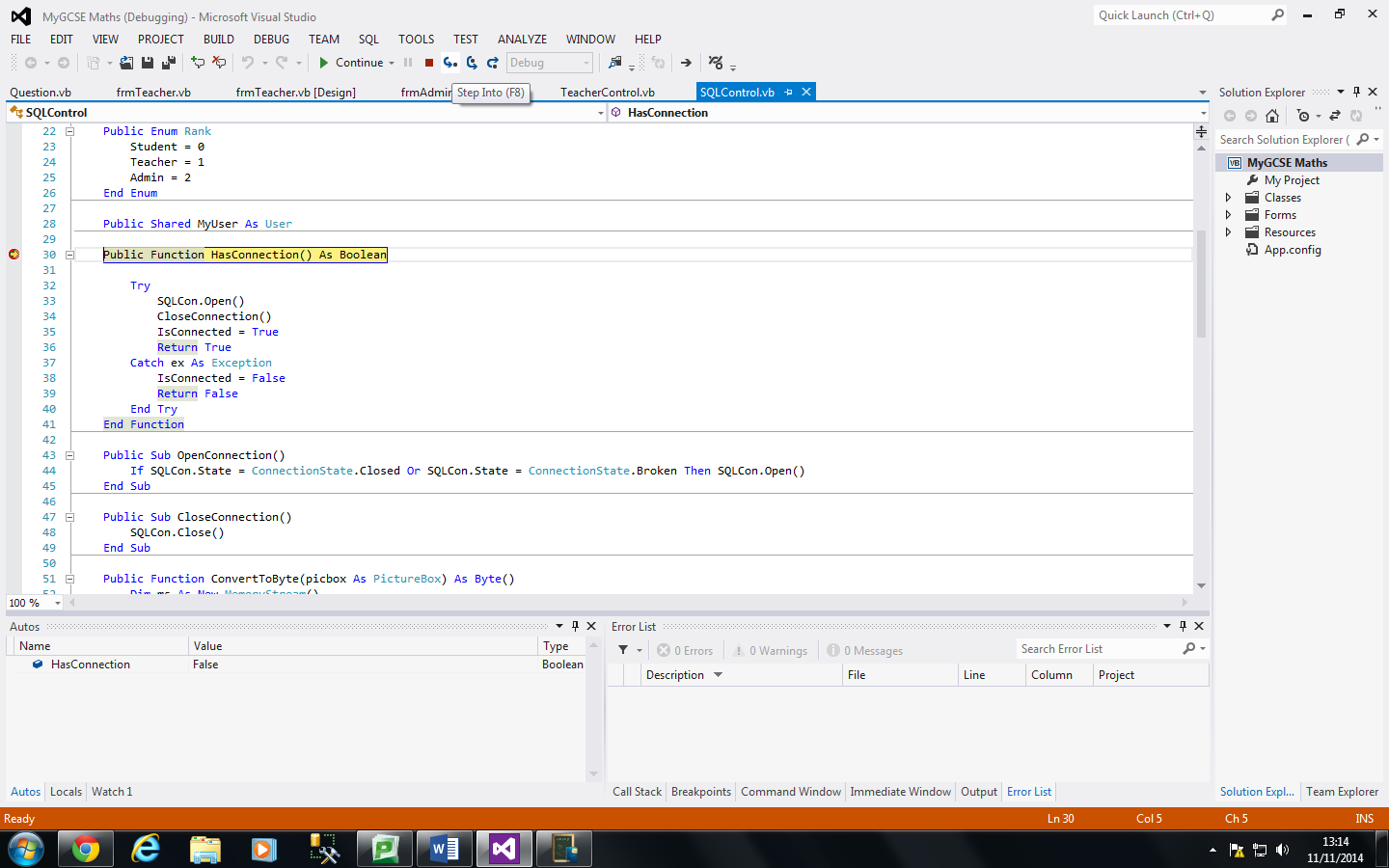
Testing Strategies

Black Box Testing

In order to complete black box testing, I will produce a test plan which includes the parts of the system I intend to test. I will then test each item in the list against the system and I will record the outcome.

White Box Testing

In order to test each route my program could go through, I will insert a breakpoint at the beginning of subroutine/method of the code I intend to test and “Step Into” each line making sure it behaves correctly. And If there is a Conditional statement e.g. If, While and Select Case. I will make sure to step into the conditional arguments too.

****

Top down testing

I will take the bottom up approach when testing. I will test the system to see if they produce the expected result. For example, for each form I have, I will test to system to ensure that it can transition from one form to another. I will also test the system to see if it can successfully generate a user’s username, password and email correctly.

Integration testing

I will also perform some integration tests. I will ensure that all parts of my system work well together. For example, I will make sure that the admin can successfully create a class, create a new teacher and student account, and then assign them to a class. I will then follow-up by ensuring that the teacher can then set quizzes for students to complete when they login to the system. Finally, I will test to see if the teacher can then view the students quiz result.

Trace Tables

For some of my algorithms like my random password generator or quick sort, I could use a trace table in order to make sure that my algorithm works properly. I will write out the value of variables upon each iteration, and if a bug occurs I would be able to fix it. However I shouldn’t be able to spot a pattern when generating passwords on the trace table because the algorithm is supposed to generate random passwords. Therefore, I would test my algorithm for creating the dynamic picture boxes in the teacher panel. I could trace each time the X and Y Location of the picture boxes change to make sure the algorithm works correctly for the right reasons.

**Test Plan**

Any text highlighted in red in the table below is an example of the data used.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test** | **Description** | **Data type** | **Expected Result** |
| 1 | Attempting to login into the system by trying to bypass the login screen using an SQL Injection technique. For example, entering a single quotation mark is generally a good way of checking for SQL vulnerability. | Erroneous  1’ OR 1 = 1 –  and  ‘ | I’ve used parameterized queries, so the system execute the login credentials literally. It should say “incorrect username or password”. |
| 2 | Able to login to the system as an Administrator. | Typical  admin  12345 | The system should direct the user to the correct form, which is the admins form. |
| 3 | The welcome label should show a welcome message to the admin. | Typical | The label to display their Firstname and Lastname along with a welcome message |
| 4 | Adding a new user of the rank admin to the system. The system should capture the UserID, Firstname, Lastname and Rank entered. | Typical  1  Jason  Mark  Admin | Message box saying user added successfully and it should add the user to the SQL database. It should also generate their username, password and email. |
| 5 | Adding a new user with an invalid userId containing a mix of strings and integers | Erroneous  48!7d7d2 | The program will say that the UserID entered is invalid and won’t add the user |
| 6 | Adding a new user to the system and for their Firstname and Lastname, I will add non alphabetic characters like &^%$£ | Erroneous  “M\*^%£  ark”  “Tho\*^ma  s-Lukasz  “£$s%” | I expect the system to remove every non-alphabetic characters (excluding – for double barrelled names) in the given string. If the input is  Sus&^%a”$n , it should return Susan |
| 7 | Adding a new user with a very long Firstname and Lastname | Erroneous | Should show an error message in the text box |
| 8 | Adding a new user with a pre-existing UserID | Erroneous  4566 | It should say UserID already exists and will not add the user |
| 9 | Adding a new class into the system | Typical  B  213 | A message box that will say Successful |
| 10 | Adding a new class (block, room), but for the room number add a character longer than 5 characters | Erroneous  F  sdfgdfgdfg | A JIT error message should be displayed since I didn’t Programmatically catch the exception. |
| 11 | Adding a class block with data longer than 1 character | Erroneous  FFF | The text box only takes 1 character. So it won’t even let me enter more than 1. |
| 12 | Using the search button to see all the classes in the system | Typical | The system should show the information of all the classes in the database, along with the teachers assigned to the classes. |
| 13 | Adding a new user of the rank teacher, and assigning that teacher to an already existing class. | Typical  8655  Rachel  Varghese  Teacher  Block B | A message box that will say Successful |
| 14 | The system should generate the email & username of the teacher accurately in line with the schema used by the college. | Typical | The system should generate the username and email of the teacher by mimicking the college’s schema for generating usernames and emails. |
| 15 | Viewing all the classes in the system | Typical | Show all classes in the system along with assigned teachers |
| 16 | Using the search feature to search for all the users in the system | Typical | A list of all the users found from the SQL database. |
| 17 | Using the search feature to search for a user that doesn’t exists. | Typical  matthew | A message box saying the user wasn’t found |
| 18 | Using the search feature to search for all users with a “an” somewhere in their name | Typical  an | A list of all the users found |
| 19 | Update the email, password and class of a user. | Typical | A message box that will say Successful |
| 19A | Update the user email address with an invalid email address. | Erroneous  hellothere  @hotmail. | The regular expression used to validate the email address will not match the email entered and thus show an error message saying that the email entered was invalid |
| 20 | Deleting a user from the system | Typical | A message box that will say Successful |
| 21 | Adding a new user of the rank student to the system, and assign the student to an existing class. | Typical  4436  Uchenna  Okafor  Block B | A message box that will say Successful |
| 22 | The system should generate the students email and username accurately, in line with the colleges schema for generating student  credentials | Typical | It should generate the email and username, and mimic the SFX college email and password scheme |
| 23 | Admin logs out of the system | Typical | Takes the user back to the login screen |
| 24 | Able to login to the system as a Teacher | Typical  rvarghese  dhgjnx21q | The system should re-direct the user to the teacher form. |
| 25 | The welcome label should welcome the teacher logged in and show the information of the class they teach | Typical | Shows teachers first name & last name along with the class they’re currently teaching. |
| 26 | Teacher searching for the students in their class. | Typical | I’ve implemented an algorithm that will dynamically display each student found onto the teacher form in a grid like effect consisting of picture boxes. For each found student I expect to see a picture box and label representation of the student. |
| 27 | The students count label to show the number of students found after the search | Typical | An Integer of the numbers of students found. |
| 28 | Teacher searching for all students whose name contain “o” | Typical  o | A display of all the students in the teachers class whose name  contain the letter “o” |
| 29 | Teacher searches for a student that doesn’t exist in their class | Typical  mary | A message box saying that no student was found |
| 30 | Attempt an SQL injection when searching for a user. | Erroneous  'Drop Table tblUsers --  AND ‘ | Since all my SQL queries are parameterized, the system should treat the malicious SQL code literally, and should say the student was not found. |
| 31 | The teacher exporting the MyGCSE Maths credentials(username and password) for each student in their class | Typical | The system should export the name, username and password of each student in the teacher’s class in the format  Name: [???]  Username: [???]  Password: [???]  into a txt file and to the chosen file path |
| 32 | Test if teacher can use the question generator form to generate 10 multi choice algebra questions of grade difficulty B. Then check if it added to the SQL table successfully. | Typical  Algebra  B  True  10 | A message box saying questions added successfully |
| 33 | Test if teacher can use the question generator form to generate 10 multi choice shapes questions of grade difficulty C. | Typical  Shapes  C  True  10 | A message box saying questions added successfully |
| 33A | After generating shapes questions, check the shape was and its labels were accurately drawn. Also, check the answer is calculated accurately. | Typical | The shape should be drawn, labelled and generated accurately. The answer to the question should be correct also. |
| 34 | Test if teacher can use the question generator form to generate 6 non multiple choice handling data questions of grade difficulty A. Then check if it added to the SQL table successfully. | Typical  Handling  Data  A  False  6 | A message box saying questions added successfully |
| 35 | The teacher adding a single choice question manually | Typical | Message box saying action was successful |
| 36 | Teacher adding a multiple choice question manually | Typical | Question added successfully |
| 37 | Teacher adding an answer that is longer than 100 characters to a question. | Erroneous | A JIT error message should be displayed since I didn’t Programmatically catch the exception. |
| 38 | Teacher searching for all the questions in the system | Typical | All the questions in the system should show in a tree view under a topic node |
| 39 | Teacher searching for all the shapes questions | Typical | Only shapes questions should show under the shape node |
| 40 | Teacher searching the system for all the B grade questions | Typical | System shows all the B grade questions |
| 41 | Teacher searching the system for all the B grade algebra questions | Typical | Should show only B grade algebra questions. |
| 42 | Teacher edits the details of a pre-existing question | Typical | Should say edit successful |
| 43 | Teacher deletes a mass amount of pre-exiting questions | Typical | Should say deleted x amount of questions successfully |
| 44 | Teacher attempts to mass delete questions without selecting any questions | Erroneous | Should show an error message saying they haven’t selected any items to delete |
| 45 | Teacher to add a set of questions to a quiz and name the quiz “AQA Maths Quiz 5” | Typical | A message box saying it was successful |
| 46 | Teacher to view all quizzes in the system | Typical | The system should show every question under each quiz. |
| 47 | Teacher to export a mark scheme of a quiz | Typical | It should export a txt file in the format  Question: [???]  Answer: [???]  In the chosen save location. |
| 48 | Teacher logs out of the system using the logout button | Typical | Logs out the system and returns back to the login form |
| 49 | Able to login as a student | Typical  Okaforu  4436  wxrslmLM | Should direct the user to the student form |
| 50 | Student attempts an uncompleted quiz that has been set by teacher | Typical | The system will prompt user if they want to start the quiz before beginning |
| 51 | The system to display a single answer question to a student | Typical | It will not show any radio buttons, only provides the user with a text box to enter the answer on. |
| 52 | The system should display a multiple choice question to a student | Typical | Each false answer and question should be displayed using radio buttons in a randomized order |
| 53 | Student answers a multiple choice question correctly | Typical  9cm^2 | Message box saying answer correct |
| 54 | Student answers a single answer question correctly | Typical  64m | Message box saying answer correct |
| 55 | Answer a question incorrectly | Typical | Message box saying answer was incorrect |
| 56 | Enter a long string as the answer of a question | Erroneous | The answer should be treated literally and should display wrong answer message box |
| 57 | When a quiz is completed the system should save the score and stores it in the SQL database | Typical | A message box showing the user their score, and saying the quiz result was saved successfully |
| 58 | The student to have no outstanding quizzes | Typical | The system should say that the student has no outstanding quizzes, and will therefore allow them to click a link to start a set of random questions. |
| 59 | Start a set of random questions – this is only possible if the student has no outstanding quizzes | Typical | The system should prompt the user before loading up questions |
| 60 | Student completing all the randomly chosen questions. | Typical | The system should show the students’ performance to the system but should not save the result. |
| 61 | While there are no questions in the database, the student should try and complete a set of random questions. | Typical | An error message should show saying that there are no questions available in the system |
| 62 | System able to log the student out of the system | Typical | The system should log the student out and take them back to the login screen |
| 63 | Login as a teacher search for a student and view their progress | Typical | Should show the students’ progress in a graph |
| 64 | Select the minimum date for when viewing student’s results. The acceptable date range is 01/10/2014 till the current date.  For example, 01/10/2014 – 24/03/2015 | Extreme | The system will show all quiz results between 01/01/2014 and the current date. |
| 65 | Selecting a time frame below 01/01/2014 | Erroneous | A message box should pop up saying that it can’t access data before 01/01/2014 |
| 66 | Export a summary of the overall quiz performance for all students in the teacher’s class. | Typical | Should export the summary in a txt file in the format  [Title]  Name: [???]  Average: [???]  Total quizzes completed: [???] |