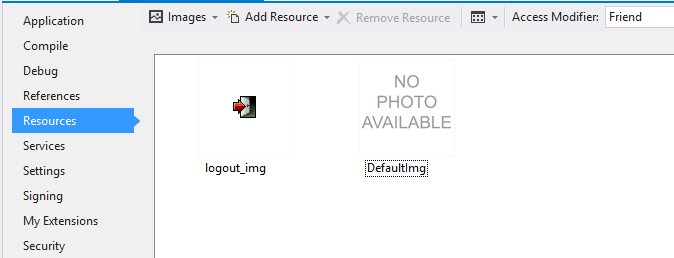
Technical Solution

Technical Solution

**Title:** Resources

|  |  |
| --- | --- |
| **Description:** | These are the resources I’ve imported. There are two image files that will be references in some parts of the project. |



**Title:** EntityBase.vb

|  |  |
| --- | --- |
| **Description:** | This is the parent class. This is the class where majority of the custom objects and shared methods are declared. This is so they can be used/referenced outside of the class. |

Imports System.Data.Sql

Imports System.Data.SqlClient

Imports System.IO

Imports System.Runtime.CompilerServices

Public Class EntityBase

Public Shared SQLCon As New SqlConnection With {.ConnectionString = "Server=sfxlabdev01\sQL2012;Database=okaforu4436\_db;Trusted\_Connection=True"}

Public Shared SQLCmd As SqlCommand

Public Shared MyUser As User

'MyUser is the user that is currently logged in to the system

#Region "Strucutres & Enum declerations"

Public Structure User

Dim UserID As Integer

Dim Firstname As String

Dim Lastname As String

Dim Username As String

Dim Password As String

Dim Rank As Rank

Dim ClassID As Integer

Dim ClassName As String

Dim Image() As Byte

Dim Email As String

'I'm overrding the ToString() method because I will be adding this Structure

'Into a ListBox control as an Object, which will run the ToString() on the item added,

'Which would return the name of the object E.g. MyGCSE\_Maths.EntityBase.User

'And by overrding the ToString() method, we can get it to show a sensible name to identify the object. Which will be the Firstname and Lastname varibles.

Overrides Function ToString() As String

Return Firstname & " " & Lastname

End Function

End Structure

Public Enum Rank

Student = 0

Teacher = 1

Admin = 2

End Enum

Public Structure Quiz

Dim QuizID As Integer

Dim Title As String

Dim Percentage As Integer

End Structure

Public Structure Question

Dim QuestionString As String

Dim QuestionID As Integer? 'The ? means it can be null

Dim Answer As String

Dim IsMultiChoice As Boolean

Dim Image As Byte()

Dim Topic As String

Dim Grade As Char

Dim FalseAnswers() As String

End Structure

Public Structure StudentQuizInfo

Dim UserID As Integer

Dim Firstname, Lastname As String

Dim QuizScores As List(Of Integer)

Dim Average As Integer 'The average of all the quiz scores

End Structure

Public Enum ActionType

Add

Edit

End Enum

Public Structure ClassRoom

Dim ClassID As Integer

Dim Block As Char 'G or A

'I'm overrding the ToString() method because I will be adding this Structure

'Into a ComboBox control as an Object, which will run the ToString() on the

item added,

'Which would return the name of the object E.g.MyGCSE\_Maths.EntityBase.ClassRoom

'And by overrding the ToString() method, we can get it to show anything we want.

Overrides Function ToString() As String

Return "Block " & Block

End Function

End Structure

Public Enum Topic

Numbers = 0

Shapes = 1

Algebra = 2

HandlingData = 3

End Enum

#End Region

#Region "Shared Methods"

Public Shared Sub OpenConnection()

'Every time I fire a new SQL query/command, this sub will be called to open

the connection.

If SQLCon.State = ConnectionState.Closed Or SQLCon.State = ConnectionState.Broken Then SQLCon.Open()

End Sub

Public Shared Sub CloseConnection()

SQLCon.Close()

End Sub

Public Overridable Sub PromptLogout()

'It will say something different for students.

If MessageBox.Show("Are you sure you want to logout?", "Prompt", MessageBoxButtons.YesNo) = DialogResult.Yes Then Logout()

End Sub

Public Sub Logout()

'Disposes all forms along with any data stored on it.

' And releases all the information stored for the current user(MyUser)

frmAdmin.Dispose()

frmEdit.Dispose()

frmStudent.Dispose()

frmTeacher.Dispose()

frmQuestions.Dispose()

frmGenerateQuestions.Dispose()

MyUser = Nothing

frmLogin.Show()

frmLogin.BringToFront()

End Sub

#End Region

End Class

**Title:** MyExtension.Module

|  |  |
| --- | --- |
| **Description:** | This is a module which contains the extension function which extends the functionality of the byte array data type and image data type. This module is inside the EntityBase class. |

Module MyExtensions

'This is a custom Extension, this extends the byte datatype and the Image object. This means I can call .ToStream()

'on any variable of the datatype Byte() same thing with an Image, I can call it like PicBox1.Image.ToBytes()

'It's just my preference, and makes calling this function look neater than doing ToBytes(PicBox1.Image)

<Extension()>

Public Function ToStream(value As Byte()) As MemoryStream

'Converts the byte array to a memory stream

If value Is Nothing Then

Return New MemoryStream(My.Resources.DefaultImg.ToBytes())

Else

Return New MemoryStream(value)

End If

End Function

<Extension()>

Public Function ToBytes(img As Image) As Byte()

'Converts the Image data/object to an array of bytes.

Dim ms As New MemoryStream

If img Is Nothing Then

My.Resources.DefaultImg.Save(ms, My.Resources.DefaultImg.RawFormat)

Else

img.Save(ms, img.RawFormat)

End If

Return ms.GetBuffer()

End Function

End Module

**Title:** CustomControls.vb

|  |  |
| --- | --- |
| **Description:** | This class contains the code for all my custom UI controls. |

Public Class CustomPictureBox

Inherits PictureBox

Private \_UserID As Integer

Private \_Username, \_Password As String

Public Property UserID As Integer

Get

Return \_UserID

End Get

Set(value As Integer)

\_UserID = value

End Set

End Property

Public Property Username As String

Get

Return \_Username

End Get

Set(value As String)

\_Username = value

End Set

End Property

Public Property Password As String

Get

Return \_Password

End Get

Set(value As String)

\_Password = value

End Set

End Property

End Class

Public Class CustomTreeNode

Inherits TreeNode

Private \_QuestionID As Integer

Public Property QuestionID() As Integer

Get

Return \_QuestionID

End Get

Set(value As Integer)

\_QuestionID = value

End Set

End Property

End Class

Public Class CustomLinkLabel

Inherits LinkLabel

Private \_QuizID As Integer

Public Property QuizID() As Integer

Get

Return \_QuizID

End Get

Set(value As Integer)

\_QuizID = value

End Set

End Property

End Class

**Title:** LoginService.vb

|  |  |
| --- | --- |
| **Description:** | This class is used for handling SQL related login requests to the system. Its methods in order to validate a users username and password in order to allow them to login or not. |

Imports System.Data.Sql

Imports System.Data.SqlClient

Public Class LoginService

Inherits EntityBase

Public IsConnected As Boolean

Public Function HasConnection() As Boolean

Try

SQLCon.Open()

SQLCon.Close()

IsConnected = True

Return True

Catch ex As SqlException

IsConnected = False

Return False

End Try

End Function

'It takes two arguments as the parameters, it will query the username, and password with the system. If it doesn’t find a match. It will return false. It it does, it will return true and set the data it finds to the MyUser variable. Which is the variable representing the user that is currently logged in.

Public Function CorrectLogin(username As String, password As String) As Boolean

OpenConnection()

Dim query As String = "SELECT \* FROM tblUser WHERE Username = @0 AND Password

= @1"

Using SQLCmd As New SqlCommand(query, SQLCon)

'We sanatize the user input to make sure it does’t contain any malicious SQL code. We paramatize our query/ use prepared statements.

SQLCmd.Parameters.AddWithValue("@0", username)

SQLCmd.Parameters.AddWithValue("@1", password)

Try

Using R As SqlDataReader = SQLCmd.ExecuteReader()

If R.HasRows Then

While R.Read

'If login found, we then set the data found to the MyUser variable.

MyUser.UserID = CInt(R("UserID"))

MyUser.Rank = CType(R("Rank"), Rank)

MyUser.Firstname = R("Firstname").ToString()

MyUser.Lastname = R("Lastname").ToString()

End While

Else

Return False

End If

End Using

If MyUser.Rank <> Rank.Admin Then

MyUser.ClassID = GetUserClassID(MyUser.UserID)

End If

Catch

Return False

End Try

End Using

CloseConnection()

Return True

End Function

Private Function GetUserClassID(userId As Integer) As Integer

'Finds and returns the ClassID of the current user.

Dim query As String = "SELECT ClassID From tblUserClass Where UserID = @0"

Using SQLCmd As New SqlCommand(query, SQLCon)

SQLCmd.Parameters.AddWithValue("@0", userId)

Using R As SqlDataReader = SQLCmd.ExecuteReader

While R.Read

Return CInt(R("ClassID"))

End While

End Using

End Using

Return 0

End Function

End Class

**Title:** QuestionGenerator.vb

|  |  |
| --- | --- |
| **Description:** | This class is used for generating different types of question. For example, shapes, algebra, numbers and handling data questions of different difficulty levels. |

Imports System.IO

Imports System.Drawing.Imaging

Imports System.Text

Imports MyGCSE\_Maths.EntityBase

Public Class QuestionGenerator

#Region "Shared Methods & Attributes"

'These are methods and attributes that I want to be shared across all the nested classes in this class.

Private Enum Difficulty

Easy

Medium

Hard

End Enum

Private Shared WordStarters() As String = {"Simplify", "Evaluate", "What is", "Work out", "What is the answer to", "Multiply out these pair of brackets"}

Private Shared Variables() As String = {"x", "y", "a", "b", "n"}

Private Shared Powers() As String = {"⁰", "½", "¹", "²"}

Private Shared Operators() As String = {"+", "-"}

Private Shared R As New Random()

Private Shared Function GetDifficulty(grade As Char) As Difficulty

'The c next to each letter is just to let the compiler know the string is infact a char variable

Select Case grade

Case "A"c To "B"c

Return Difficulty.Hard

Case "C"c To "D"c

Return Difficulty.Medium

Case "E"c

Return Difficulty.Easy

End Select

Return Nothing

End Function

Private Shared Function GetRandomPower() As String

Dim index As Integer = R.Next(0, Powers.Length)

Return Powers(index)

End Function

Private Shared Function GetRandomWord() As String

Dim index As Integer = R.Next(0, WordStarters.Length - 1) 'I intentionally didn't want to get the last word.

Return WordStarters(index)

End Function

Private Shared Function GetRandomOperator() As String

Dim index As Integer = R.Next(0, Operators.Length)

Return Operators(index)

End Function

Private Shared Function GetRandomVariable() As String

Dim index As Integer = R.Next(0, Variables.Length)

Return Variables(index)

End Function

Private Shared Function GetRandomNumber(difficultyLvl As Difficulty) As Integer

Select Case difficultyLvl

Case Difficulty.Easy

Return R.Next(2, 10)

Case Difficulty.Medium

Return R.Next(10, 30)

Case Difficulty.Hard

Return R.Next(15, 50)

End Select

Return Nothing

End Function

#End Region

Public Shared Function GenerateQuestions(qTopic As Topic, grade As Char, amountRequested As Integer, multiChoice As Boolean) As List(Of Question)

Dim difficultyLevel As Difficulty = GetDifficulty(grade)

Dim questionList As New List(Of Question)

For i = 1 To amountRequested

Dim question As New Question

Select Case qTopic

Case Topic.Algebra

question = New AlgebraQuestion().GenerateQuestion()

Case Topic.Numbers

question = New NumbersQuestion().GenerateQuestion(difficultyLevel)

Case Topic.Shapes

question = New ShapesQuestion().GenerateQuestion()

Case Topic.HandlingData

question = New HandlingDataQuestion().GenerateQuestion(difficultyLevel)

End Select

question.Grade = grade

question.IsMultiChoice = multiChoice

questionList.Add(question)

Next

Return questionList

End Function

#Region "Question Generator Classes"

Private Class AlgebraQuestion

Private thisQuestion As New Question With {.Image = My.Resources.DefaultImg.ToBytes(), .Topic = "Algebra"}

Private Structure Bracket '(x+3)

Dim Variable As String 'e.g x or y

Dim [Operator] As String 'e.g - or +

Dim Number As Integer 'e.g 3

End Structure

Public Function GenerateQuestion() As Question

Dim word As String = WordStarters(R.Next(2, WordStarters.Length)) 'I only wanted certain values

Dim var As Char = CChar(GetRandomVariable())

Dim bracket1 As New Bracket With {.Variable = var, .Number = R.Next(2, 15), .Operator = GetRandomOperator()}

Dim bracket2 As New Bracket With {.Variable = var, .Number = R.Next(2, 15), .Operator = GetRandomOperator()}

thisQuestion.QuestionString = String.Format("{0} ({1} {2} {3})({4} {5} {6})", word, bracket1.Variable, bracket1.Operator, bracket1.Number, bracket2.Variable, bracket2.Operator, bracket2.Number)

'E.g. Calculate (x + 5)(x + 7)

Dim firstBracketNum As Integer = CInt(bracket1.Operator & bracket1.Number) ' e.g -5

Dim secondBracketNum As Integer = CInt(bracket2.Operator & bracket2.Number) ' e.g +8

thisQuestion.Answer = SolveQuestion(var, firstBracketNum, secondBracketNum)

thisQuestion.FalseAnswers = GenerateFalseAnswers(var, firstBracketNum, secondBracketNum)

Return thisQuestion

End Function

Private Function SolveQuestion(variable As Char, firstBracket As Integer, secondBracket As Integer) As String

Dim sb As New StringBuilder

'ax² + bx + c 'This is how the format/forumla of what the answer would be like.

Dim b As Integer = firstBracket + secondBracket 'The numbers in the first & second bracket. (x+1)(x+3) so the 1 and 3

Dim c As Integer = firstBracket \* secondBracket

sb.Append(variable & "²") ' e.g. b²

If b > 0 Then

sb.Append(" + " & b & variable)

ElseIf b < 0 Then

sb.Append(" " & b & variable)

End If

' e.g. b² + 5b

If c > 0 Then

sb.Append(" + " & c)

ElseIf c < 0 Then

sb.Append(" " & c)

End If

' e.g. b² + 5b - 6

Return sb.ToString()

End Function

Private Function GenerateFalseAnswers(var As Char, firstBracket As Integer, secondBracket As Integer) As String()

Dim sb As New StringBuilder

Dim answersFalse(2) As String

'ax² + bx + c 'This is how the format/forumla of what the answer would be like.

firstBracket += R.Next(-2, 5) 'Just to make it random so we can recieve a random false answer

For i = 0 To 2

sb.Clear()

Dim b As Integer = firstBracket + Not secondBracket - (R.Next(-5, 5)) 'Just to make it random so we can recieve a random false answer

Dim c As Integer = Not firstBracket \* secondBracket + (R.Next(-5, 5)) 'Just to make it random so we can recieve a random false answer

sb.Append(var & "²")

If b > 0 Then

sb.Append(" + " & b & var)

ElseIf b < 0 Then

sb.Append(" " & b & var)

End If

If c > 0 Then

sb.Append(" + " & c)

ElseIf c < 0 Then

sb.Append(" " & c)

End If

answersFalse(i) = sb.ToString

Next

Return answersFalse

End Function

End Class

Private Class NumbersQuestion

Private thisQuestion As New Question With {.Image = My.Resources.DefaultImg.ToBytes(), .Topic = "Numbers"}

Public Function GenerateQuestion(difficultyLevel As Difficulty) As Question

Dim number As Integer = GetRandomNumber(difficultyLevel)

Dim power As String = GetRandomPower()

thisQuestion.QuestionString = String.Format("{0} {1}{2}", GetRandomWord, number, power) 'e.g {Evaluate} {5}{²}

thisQuestion.Answer = SolveQuestion(CInt(number), power)

thisQuestion.FalseAnswers = GenerateFalseAnswers(thisQuestion.Answer)

Return thisQuestion

End Function

Private Function SolveQuestion(number As Integer, power As String) As String

Select Case power

Case "⁰"

Return Math.Pow(number, 0).ToString

Case "½"

Return (number / 2).ToString

Case "¹"

Return Math.Pow(number, 1).ToString

Case "²"

Return Math.Pow(number, 2).ToString

Case Else

Return Nothing

End Select

End Function

Private Function GenerateFalseAnswers(correctAnswer As String) As String()

Dim falseAnswers(2) As String

falseAnswers(0) = (CInt(correctAnswer) + R.Next(1, 5)).ToString

falseAnswers(1) = (CInt(correctAnswer) - R.Next(1, 10)).ToString

falseAnswers(2) = (CInt(correctAnswer) \* R.Next(2, 5)).ToString

Return falseAnswers

End Function

End Class

Private Class ShapesQuestion

Private Structure TPoint

Dim Text As String

Dim xLocation As Integer

Dim yLocation As Integer

Dim Format As StringFormatFlags

End Structure

Private Enum Shape

Square

Rectangle

Triangle

End Enum

Private Enum QuestionType

Perimeter

Area

End Enum

Dim Shapes() As Shape = {Shape.Rectangle, Shape.Square, Shape.Triangle}

Private QTypes() As QuestionType = {QuestionType.Area, QuestionType.Perimeter}

Private Measurements() As String = {"cm", "m"}

Private thisQuestion As New Question With {.Topic = "Shapes"}

Public Function GenerateQuestion() As Question

ReDim thisQuestion.FalseAnswers(2)

Dim rndShape As Shape = Shapes(R.Next(0, Shapes.Length)) 'Gets a random shape from an array of shapes.

Dim questionType As QuestionType = QTypes(R.Next(0, QTypes.Length)) 'Gets random question type. Perimeter or Area

Dim m As String = Measurements(R.Next(0, Measurements.Length)) ' cm or m

Select Case rndShape

' It will also solve the question, and the false answers for the questions too

Case Shape.Rectangle

thisQuestion.Image = Image.FromStream(DrawRectangle(questionType, m)).ToBytes()

Case Shape.Square

thisQuestion.Image = Image.FromStream(DrawSquare(questionType, m)).ToBytes()

Case Shape.Triangle

thisQuestion.Image = Image.FromStream(DrawTriangle(questionType, m)).ToBytes()

End Select

Dim a() As String = {"Work out the", "What is the", "Calculate the"}

Dim b As String = a(R.Next(0, a.Length)) ' Picks random word from the a() array

thisQuestion.QuestionString = String.Format("{0} {1} of this {2}", b, questionType.ToString.ToLower, rndShape.ToString)

'E.g {Work out the} {area} of this {Triangle}

Return thisQuestion

End Function

Private Function DrawTriangle(type As QuestionType, unit As String) As MemoryStream

Dim hypot As Integer = 0

Dim adjac As Integer = 0

Dim oppos As Integer = 0

' These are the numeric values that will be plotted on each side of the triangle

Do

hypot = R.Next(5, 20)

adjac = R.Next(3, 15)

oppos = R.Next(2, 10)

Loop Until hypot > adjac And oppos < adjac

' This is to make sure the Adjacent and Opposite sides are never greater than

' the hypotenouse. Therefore generating a more realistic question

Dim lblHypotenouse As New TPoint With {.Text = hypot & unit, .xLocation = 140, .yLocation = 100}

Dim lblAdjacent As New TPoint With {.Text = adjac & unit, .xLocation = 20, .yLocation = 115}

Dim lblOpposite As New TPoint With {.Text = oppos & unit, .xLocation = 115, .yLocation = 216}

Dim coordinateCollection As New Collection From {lblHypotenouse, lblAdjacent, lblOpposite}

'add to collection so I can take advantage of the for each loop

Dim canvas As New Bitmap(260, 260)

Dim myFont As New Font("Arial", 10)

Dim points(2) As Point ' Each point of the triangle I am yet to draw

points(0) = New Point(58, 29)

points(1) = New Point(58, 212)

points(2) = New Point(214, 212)

Using g As Graphics = Graphics.FromImage(canvas) : g.Clear(Color.White)

g.DrawPolygon(Pens.Red, points) ' Draws Triangle

For Each point As TPoint In coordinateCollection

g.DrawString(point.Text, myFont, Brushes.Green, New Point(point.xLocation, point.yLocation))

'Draws labels on the triangle

Next

g.Save()

End Using

SolveTriangleQuestion(type, hypot, oppos, adjac, unit)

Dim memoryStrm As New MemoryStream()

canvas.Save(memoryStrm, ImageFormat.Jpeg)

DisposeObjects(canvas, myFont)

Return memoryStrm

End Function

Private Function DrawSquare(type As QuestionType, unit As String) As MemoryStream

'unit. E.g. cm or m

Dim num As Integer = R.Next(1, 50)

Dim labelA As New TPoint With {.Text = num & unit, .xLocation = 110, .yLocation = 15}

Dim labelB As New TPoint With {.Text = num & unit, .xLocation = 15, .yLocation = 110, .Format = StringFormatFlags.DirectionVertical}

Dim coordinateCollection As New Collection From {labelA, labelB}

Dim canvas As New Bitmap(260, 260)

Dim myFont As New Font("Arial", 10)

Using g As Graphics = Graphics.FromImage(canvas) : g.Clear(Color.White)

g.DrawRectangle(Pens.Blue, 35, 35, 190, 190) ' Draws the Square

For Each point As TPoint In coordinateCollection

Dim sF As New StringFormat With {.FormatFlags = point.Format} 'This allows me to rotate the text

g.DrawString(point.Text, myFont, Brushes.Red, New Point(point.xLocation, point.yLocation), sF)

Next

g.Save()

End Using

SolveSquareQuestion(type, num, unit)

Dim memoryStrm As New MemoryStream()

canvas.Save(memoryStrm, ImageFormat.Jpeg)

DisposeObjects(canvas, myFont)

Return memoryStrm

End Function

Private Function DrawRectangle(type As QuestionType, unit As String) As MemoryStream

Dim base As Integer = R.Next(5, 55)

Dim height As Integer = R.Next(5, 25)

Do

base = R.Next(5, 55)

height = R.Next(5, 25)

Loop Until base > height ' Ensures the base length is always greater

Dim lblBase As New TPoint With {.Text = base & unit, .xLocation = 108, .yLocation = 30}

Dim lblHeight As New TPoint With {.Text = height & unit, .xLocation = 15, .yLocation = 100}

Dim coordinateCollection As New Collection From {lblBase, lblHeight}

Dim canvas As New Bitmap(260, 260)

Dim myFont As New Font("Arial", 10)

Using g As Graphics = Graphics.FromImage(canvas) : g.Clear(Color.White)

g.DrawRectangle(Pens.Blue, 15, 50, 230, 120) ' Draws the rectangle

For Each point As TPoint In coordinateCollection

g.DrawString(point.Text, myFont, Brushes.Red, New Point(point.xLocation, point.yLocation))

Next 'Draws the text of each label on each side.

g.Save()

End Using

SolveRectangleQuestion(type, base, height, unit)

Dim memoryStrm As New MemoryStream()

canvas.Save(memoryStrm, ImageFormat.Jpeg)

DisposeObjects(canvas, myFont)

Return memoryStrm

End Function

Private Sub SolveTriangleQuestion(type As QuestionType, hypo As Integer, opposite As Integer, adjacent As Integer, unit As String)

If type = QuestionType.Area Then

thisQuestion.Answer = ((opposite \* adjacent) / 2) & unit & "²" ' Calculates the area

thisQuestion.FalseAnswers(0) = (hypo + opposite + adjacent) & unit ' Confuses student by calculating perimeter

thisQuestion.FalseAnswers(1) = (opposite \* adjacent) & unit & "²"

ElseIf type = QuestionType.Perimeter Then

thisQuestion.Answer = (hypo + opposite + adjacent) & unit ' Calculates the perimeter

thisQuestion.FalseAnswers(0) = ((opposite \* adjacent) / 2) & unit & "²" ' Confuses student by calculating area

thisQuestion.FalseAnswers(1) = hypo + opposite & unit

End If

thisQuestion.FalseAnswers(2) = R.Next(opposite, hypo) & unit

End Sub

Private Sub SolveSquareQuestion(type As QuestionType, side As Integer, unit As String)

If type = QuestionType.Area Then

thisQuestion.Answer = Math.Pow(side, 2) & unit & "²"

thisQuestion.FalseAnswers(0) = (side \* 4) & unit ' To throw off the student

ElseIf type = QuestionType.Perimeter Then

thisQuestion.Answer = (side \* 4) & unit

thisQuestion.FalseAnswers(0) = Math.Pow(side, 2) & unit & "²" ' To throw off the student

End If

thisQuestion.FalseAnswers(1) = ((side / 2) + R.Next(0, side)) & unit

thisQuestion.FalseAnswers(2) = ((side / 2) \* 4) & unit

End Sub

Private Sub SolveRectangleQuestion(type As QuestionType, base As Integer, height As Integer, unit As String)

If type = QuestionType.Area Then

thisQuestion.Answer = base \* height & unit & "²"

thisQuestion.FalseAnswers(0) = (base \* 2) + (height \* 2) & unit ' To throw off the student

ElseIf type = QuestionType.Perimeter Then

thisQuestion.Answer = (base \* 2) + (height \* 2) & unit

thisQuestion.FalseAnswers(0) = base \* height & unit & "²" ' To throw off the student

End If

thisQuestion.FalseAnswers(1) = R.Next(height, base) & unit

thisQuestion.FalseAnswers(2) = R.Next(height, base) & unit & "²"

End Sub

Private Sub DisposeObjects(ByRef b As Bitmap, ByRef f As Font)

b.Dispose()

f.Dispose()

End Sub

End Class

Private Class HandlingDataQuestion

Private thisQuestion As New Question With {.Image = My.Resources.DefaultImg.ToBytes(), .Topic = "Handling Data"}

Public Function GenerateQuestion(difficultyLevel As Difficulty) As Question

Dim percentages() As Integer = {5, 10, 12, 15, 17, 20, 22, 25, 30, 32, 35, 40, 42, 50} 'the list of possible percentages

Dim number As Integer = GetRandomNumber(difficultyLevel)

Dim percent As Integer = percentages(R.Next(0, percentages.Length)) 'gets a random percentage from the percentages() array

Dim a() As String = {"What is", "Work out", "Calculate the answer to", "Work out the percentage to"}

Dim b As String = a(R.Next(0, a.Length)) 'Gets a random word from the array above

thisQuestion.QuestionString = String.Format("{0} {1}% of £{2}", b, percent, number)

thisQuestion.Answer = SolveQuestion(percent, number)

thisQuestion.FalseAnswers = GenerateFalseAnswers(CDbl(thisQuestion.Answer))

Return thisQuestion

End Function

Private Function SolveQuestion(percentage As Integer, number As Integer) As String

Dim onePercent As Double = percentage / 100 'Find out what one percent is first.

Return "£" & (onePercent \* number).ToString 'Then times it by the amount of percentage we need

End Function

Private Function GenerateFalseAnswers(answer As Double) As String()

Dim falseAnswers(2) As String

falseAnswers(0) = "£" & (answer + R.Next(2, 10)).ToString

falseAnswers(1) = "£" & (answer - R.Next(2, 5)).ToString

falseAnswers(2) = "£" & (answer \* R.Next(2, 5)).ToString

Return falseAnswers

End Function

End Class

#End Region

End Class

**Title:** QuestionHelper.vb

|  |  |
| --- | --- |
| **Description:** | This is a shared/static class which is used to manage the questions in the SQL database. This class can be used for adding, deleting, editing and fetching questions. |

Imports System.Data.SqlClient

Imports MyGCSE\_Maths.EntityBase

Public Class QuestionHelper

Public Shared Function AddQuestion(question As Question) As Boolean

OpenConnection()

Dim query As String = "Insert Into tblQuestion(Topic, Grade, Question, Image, Answer, MultiChoice) Values(@0, @1, @2, @3, @4, @5);"

Using SQLCmd As New SqlCommand(query, SQLCon)

SQLCmd.Parameters.AddWithValue("@0", question.Topic)

SQLCmd.Parameters.AddWithValue("@1", question.Grade)

SQLCmd.Parameters.AddWithValue("@2", question.QuestionString)

SQLCmd.Parameters.AddWithValue("@3", question.Image)

SQLCmd.Parameters.AddWithValue("@4", question.Answer)

SQLCmd.Parameters.AddWithValue("@5", question.IsMultiChoice)

If SQLCmd.ExecuteNonQuery = 1 Then

If question.IsMultiChoice = True Then

Return AddFalseAnswers(question.FalseAnswers)

Else

Return True

End If

Else

Return False

End If

End Using

CloseConnection()

Return Nothing

End Function

Public Shared Function DeleteQuestion(questionId As Integer) As Boolean

OpenConnection()

Dim query As String = "Delete From tblQuestion Where QID = @0;" \_

& "Delete From tblWrongAnswers Where QID = @0;" \_

& "Delete From tblQuiz Where QID = @0"

Using SQLCmd As New SqlCommand(query, SQLCon)

SQLCmd.Parameters.AddWithValue("@0", questionId)

If SQLCmd.ExecuteNonQuery >= 1 Then

Return True

Else

Return False

End If

End Using

End Function

Public Shared Function UpdateQuestion(QuestionID As Integer, newQuestion As Question) As Boolean

OpenConnection()

Dim query As String = "Update tblQuestion Set Topic = @0, Grade = @1, Question = @2, Image = @3, Answer = @4, MultiChoice = @5 Where QID = @6"

Using SQLCmd As New SqlCommand(query, SQLCon)

SQLCmd.Parameters.AddWithValue("@0", newQuestion.Topic)

SQLCmd.Parameters.AddWithValue("@1", newQuestion.Grade)

SQLCmd.Parameters.AddWithValue("@2", newQuestion.QuestionString)

SQLCmd.Parameters.AddWithValue("@3", newQuestion.Image)

SQLCmd.Parameters.AddWithValue("@4", newQuestion.Answer)

SQLCmd.Parameters.AddWithValue("@5", newQuestion.IsMultiChoice)

SQLCmd.Parameters.AddWithValue("@6", QuestionID)

If SQLCmd.ExecuteNonQuery = 1 Then

If newQuestion.IsMultiChoice = True Then

Return UpdateMultiChoice(QuestionID, newQuestion.FalseAnswers)

Else

Return True

End If

Else

Return False

End If

End Using

CloseConnection()

End Function

Private Shared Function UpdateMultiChoice(questionId As Integer, falseAnswers() As String) As Boolean

OpenConnection()

Dim query As String = "Update tblWrongAnswers Set Dummy1 = @A1, Dummy2 = @A2, Dummy3 = @A3 Where QID = @QID"

Using SQLCmd As New SqlCommand(query, SQLCon)

SQLCmd.Parameters.AddWithValue("@QID", questionId)

SQLCmd.Parameters.AddWithValue("A1", falseAnswers(0))

SQLCmd.Parameters.AddWithValue("A2", falseAnswers(1))

SQLCmd.Parameters.AddWithValue("A3", falseAnswers(2))

If SQLCmd.ExecuteNonQuery = 1 Then

Return True

Else

Return False

End If

End Using

CloseConnection()

End Function

Private Shared Function GetNextIncrement() As Integer

Dim query As String = "Select QID From tblQuestion Order By QID Desc"

Dim \_NextID As Integer = 1

'Setting it as one because if there were

'no records, the next one would be 1 since my Auto Increment increments by 1

Using SQLCmd As New SqlCommand(query, SQLCon)

\_NextID = CInt(SQLCmd.ExecuteScalar)

End Using

Return \_NextID

End Function

Private Shared Function AddFalseAnswers(wrongAnswers() As String) As Boolean

Dim query As String = "Insert Into tblWrongAnswers(QID, Dummy1, Dummy2, Dummy3) Values(@0, @1, @2, @3)"

Using SQLCmd As New SqlCommand(query, SQLCon)

SQLCmd.Parameters.AddWithValue("@0", GetNextIncrement)

SQLCmd.Parameters.AddWithValue("@1", wrongAnswers(0))

SQLCmd.Parameters.AddWithValue("@2", wrongAnswers(1))

SQLCmd.Parameters.AddWithValue("@3", wrongAnswers(2))

If SQLCmd.ExecuteNonQuery = 1 Then

Return True

Else

Return False

End If

End Using

End Function

Public Shared Function GetQuestion(questionId As Integer) As Question

OpenConnection()

'Gets all the information of a question and returns the question

Dim query As String = "Select \* From tblQuestion Where QID = @0"

Dim q As New Question

Using SQLCmd As New SqlCommand(query, SQLCon)

SQLCmd.Parameters.AddWithValue("@0", questionId)

Using R As SqlDataReader = SQLCmd.ExecuteReader

While R.Read

q.QuestionID = CInt(R("QID"))

q.QuestionString = R("Question").ToString()

q.Image = CType(R("Image"), Byte())

q.Answer = R("Answer").ToString()

q.IsMultiChoice = CBool(R("MultiChoice"))

q.Topic = R("Topic").ToString()

q.Grade = CChar(R("Grade"))

End While

End Using

End Using

If q.IsMultiChoice Then q.FalseAnswers = GetFalseAnswers(CInt(q.QuestionID))

CloseConnection()

Return q

End Function

Private Shared Function GetFalseAnswers(questionId As Integer) As String()

OpenConnection()

Dim query As String = "Select \* From tblWrongAnswers Where QID = @0"

Dim answersArray(3) As String

'Creates an array, and which all the fetched wrong answers will be added to.

Using SQLCmd As New SqlCommand(query, SQLCon)

SQLCmd.Parameters.AddWithValue("@0", questionId)

Using R As SqlDataReader = SQLCmd.ExecuteReader

While R.Read

answersArray(0) = R("Dummy1").ToString()

answersArray(1) = R("Dummy2").ToString()

answersArray(2) = R("Dummy3").ToString()

End While

End Using

End Using

CloseConnection()

'Returns all the false answers found.

Return answersArray

End Function

End Class

|  |  |
| --- | --- |
| **Description:** | This class is used to manage the users and classes in the SQL database. The methods in this class helps the admin perform backend tasks. For example, adding or deleting a user from the system. |

**Title:** AdminService.vb

Imports System.Data.Sql

Imports System.Data.SqlClient

Imports System.Text

Imports System.Text.RegularExpressions

Public Class AdminService

Inherits EntityBase

#Region "frmAdmin"

Public Function GetUsers(name As String) As List(Of User)

OpenConnection()

'Searches for all the students' firstname from the sql user table

Dim query As String = "SELECT \* FROM tblUser " \_

& "WHERE Firstname + ' ' + Lastname LIKE @0 " \_

& "Order By Rank Desc"

Dim usersList As New List(Of User)

Using SQLCmd As New SqlCommand(query, SQLCon)

SQLCmd.Parameters.AddWithValue("@0", "%" & name & "%")

Using R As SqlDataReader = SQLCmd.ExecuteReader()

If R.HasRows Then

While R.Read

Dim thisUser As New User

thisUser.UserID = CInt(R("UserID"))

thisUser.Firstname = R("Firstname").ToString

thisUser.Lastname = R("Lastname").ToString

thisUser.Rank = CType(R("Rank"), Rank)

'adds all the users found to a usersList.

usersList.Add(thisUser)

End While

Else

Return Nothing

End If

End Using

End Using

CloseConnection()

'Returns all the users found

Return usersList

End Function

Public Function AddAccount(userId As Integer, firstname As String, lastname As String, userRank As Rank, img As Byte(), classId As Integer) As Boolean

OpenConnection()

Dim query As String = "INSERT INTO tblUser(UserID,Firstname,Lastname,Email,Picture,Username,Password,Rank) " \_

& "VALUES (@0, @1, @2, @3, @img, @4, @5, @6);"

'removes every occurance of a non alphabetic character

firstname = ParseStr(firstname)

lastname = ParseStr(lastname)

Dim username As String = GenerateUsername(userId, firstname, lastname, userRank)

Dim password As String = GeneratePassword()

Dim email As String = GenerateEmail(userId, firstname, lastname, userRank)

Try

Using SQLCmd As New SqlCommand(query, SQLCon)

SQLCmd.Parameters.AddWithValue("@0", userId)

SQLCmd.Parameters.AddWithValue("@1", firstname)

SQLCmd.Parameters.AddWithValue("@2", lastname)

SQLCmd.Parameters.AddWithValue("@3", email)

SQLCmd.Parameters.AddWithValue("@img", img)

SQLCmd.Parameters.AddWithValue("@4", username)

SQLCmd.Parameters.AddWithValue("@5", password)

SQLCmd.Parameters.AddWithValue("@6", CInt(userRank))

If SQLCmd.ExecuteNonQuery = 1 Then

If Not userRank = Rank.Admin Then

Return AddUserClass(userId, classId)

Else

Return True

End If

Else

Return False

End If

End Using

Catch ex As SqlException

If ex.Message.Contains("Cannot insert duplicate key") Then

Throw New Exception("The UserID: " & userId & " already exists")

Else

Throw New Exception(ex.Message)

End If

End Try

CloseConnection()

End Function

Private Function AddUserClass(userId As Integer, classId As Integer) As Boolean

'Adds a user to a class. Only teachers and students can be added to a class.

Dim query As String = "Insert Into tblUserClass(UserID, ClassID) Values(@0, @1);"

Using SQLCmd As New SqlCommand(query, SQLCon)

SQLCmd.Parameters.AddWithValue("@0", userId)

SQLCmd.Parameters.AddWithValue("@1", classId)

If SQLCmd.ExecuteNonQuery = 1 Then

Return True

Else

Return False

End If

End Using

End Function

Private Function ParseStr(txt As String) As String

' Replaces every occurance of a non alphabetic text, excluding - and returns the value

Return Regex.Replace(txt, "[^a-zA-Z\- ]", String.Empty)

End Function

Private Function GenerateEmail(userId As Integer, firstname As String, lastname As String, userRank As Rank) As String

'Generates the Email of the user according to the information provided through the paramaters.

'This mimics the college system

Dim email As String = String.Empty

If userRank = Rank.Student Then

If lastname.Length > 7 Then lastname = lastname.Substring(0, 7)

email = String.Format("{0}{1}{2}@student.sfx.ac.uk", lastname, firstname.Substring(0, 1), userId)

Else

email = String.Format("{0}.{1}@sfx.ac.uk", firstname.Substring(0, 1), lastname)

End If

Return email.ToLower()

End Function

Private Function GenerateUsername(userId As Integer, firstname As String, lastname As String, userRank As Rank) As String

Dim username As String = String.Empty

If userRank = Rank.Student Then

If lastname.Length > 7 Then lastname = lastname.Substring(0, 7)

username = lastname & firstname.Substring(0, 1) & userId.ToString

Else

username = firstname.Substring(0, 1) & lastname

End If

Return username.ToLower()

End Function

Private Function GeneratePassword() As String

Dim str As String = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"

Dim r As New Random

Dim sb As New StringBuilder

For i = 1 To 4

Dim rdnNum As Integer = r.Next(0, str.Length - 1)

sb.Append(str.Substring(rdnNum, 2))

' Picks a random substring from the str variable and appends it to a string builder

Next

Return sb.ToString()

End Function

Public Function GetAllClasses() As List(Of ClassRoom)

OpenConnection()

Dim classList As New List(Of ClassRoom)

Dim query As String = "Select \* From tblClass Order By Block Asc"

'Selects all the classes in the system

Using SQLCmd As New SqlCommand(query, SQLCon)

Using R As SqlDataReader = SQLCmd.ExecuteReader()

While R.Read

Dim block As Char = CChar(R("Block"))

Dim classId As Integer = CInt(R("ClassID"))

Dim item = New ClassRoom With {.Block = block, .ClassID = classId}

classList.Add(item)

End While

End Using

End Using

CloseConnection()

Return classList

End Function

Public Function CreateClass(block As Char, room As String) As Boolean

OpenConnection()

'Creates a new class

Dim query As String = "Insert Into tblClass(Block,Room) Values(@0, @1)"

Using SQLCmd As New SqlCommand(query, SQLCon)

SQLCmd.Parameters.AddWithValue("@0", block)

SQLCmd.Parameters.AddWithValue("@1", room)

If SQLCmd.ExecuteNonQuery = 1 Then

Return True

Else

Return False

End If

End Using

CloseConnection()

End Function

Public Sub AddOccupiedClasses()

OpenConnection()

'Gets information of what class either a teacher or students belongs to

Dim query As String = "Select Block,Room,Firstname,Lastname From tblClass,tblUserClass,tblUser " \_

& "Where Rank = 1 And tblUser.UserID = tblUserClass.UserID " \_

& "And tblUserClass.ClassID = tblClass.ClassID "

SQLCmd = New SqlCommand(query, SQLCon)

Using R As SqlDataReader = SQLCmd.ExecuteReader()

While R.Read

frmAdmin.DataGridView1.Rows.Add({R("Block").ToString(), R("Room").ToString(), R("Firstname").ToString(), R("Lastname").ToString()})

'Adds each array of item fetched to the DataGridView

End While

End Using

CloseConnection()

End Sub

Public Sub AddEmptyClasses()

'Gets all the classes that no teacher is currently teaching

OpenConnection()

Dim query As String = "Select \* From tblClass " \_

& "Where tblClass.ClassID NOT IN " \_

& "(Select ClassID From tblUserClass, tblUser Where tblUser.UserID = tblUserClass.UserID And tblUser.[Rank] = 1)"

Using SQLCmd = New SqlCommand(query, SQLCon)

Using R As SqlDataReader = SQLCmd.ExecuteReader()

While R.Read

frmAdmin.DataGridView1.Rows.Add({R("Block").ToString(), R("Room").ToString()})

End While

End Using

End Using

CloseConnection()

End Sub

#End Region

#Region "frmEdit"

Public Function DeleteAccount(userId As Integer) As Boolean

OpenConnection()

Dim query As String = "DELETE FROM tblUser Where UserID = @0;" \_

& "DELETE FROM tblQuizLog Where UserID = @0;" \_

& "DELETE FROM tblUserClass Where UserID = @0;"

Try

SQLCmd = New SqlCommand(query, SQLCon)

SQLCmd.Parameters.AddWithValue("@0", userId)

If SQLCmd.ExecuteNonQuery >= 1 Then

Return True

Else

Return False

End If

Catch ex As SqlException

Throw New Exception(ex.Message)

End Try

CloseConnection()

End Function

Public Function UpdateAccount(userId As Integer, firstname As String, lastname As String, email As String, password As String, img As Byte(), classId As Integer, userRank As Rank) As Boolean

OpenConnection()

Dim query As String = String.Empty

If userRank = Rank.Admin Then

query = "UPDATE tblUser SET Firstname = @1, Lastname = @2, Email = @3, Picture = @4, Password = @5 WHERE UserID = @0;"

Else

query = "UPDATE tblUser SET Firstname = @1, Lastname = @2, Email = @3, Picture = @4, Password = @5 WHERE UserID = @0;" \_

& " UPDATE tblUserClass SET ClassID = @6 WHERE UserID = @0"

End If

Try

SQLCmd = New SqlCommand(query, SQLCon)

SQLCmd.Parameters.AddWithValue("@0", userId)

SQLCmd.Parameters.AddWithValue("@1", firstname)

SQLCmd.Parameters.AddWithValue("@2", lastname)

SQLCmd.Parameters.AddWithValue("@3", email)

SQLCmd.Parameters.AddWithValue("@4", img)

SQLCmd.Parameters.AddWithValue("@5", password)

If userRank <> Rank.Admin Then SQLCmd.Parameters.AddWithValue("@6", classId)

If SQLCmd.ExecuteNonQuery >= 1 Then

Return True

Else

Return False

End If

Catch ex As SqlException

Throw New Exception(ex.Message)

End Try

CloseConnection()

End Function

Public Function GetUserInformation(userId As Integer, userRank As Rank) As User

OpenConnection()

Dim query As String = String.Empty

Dim user As New User

If userRank = Rank.Admin Then

query = "SELECT \* From tblUser WHERE tblUser.UserID = @0"

Else

query = "SELECT \* From tblUser, tblClass, tblUserClass " \_

& "WHERE tblUserClass.ClassID = tblClass.ClassID " \_

& "AND tblUser.UserID = tblUserClass.UserID AND tblUser.UserID = @0"

End If

SQLCmd = New SqlCommand(query, SQLCon)

SQLCmd.Parameters.AddWithValue("@0", userId)

Using R As SqlDataReader = SQLCmd.ExecuteReader()

While R.Read

user.UserID = CInt(R("UserID"))

user.Username = R("Username").ToString()

user.Password = R("Password").ToString()

user.Rank = CType(R("Rank"), Rank)

If userRank <> Rank.Admin Then user.ClassName = "Block " & R("Block").ToString ' E.g. Block A

user.Firstname = R("Firstname").ToString()

user.Lastname = R("Lastname").ToString()

user.Email = R("Email").ToString()

user.Image = CType(R("Picture"), Byte())

End While

End Using

CloseConnection()

Return user

End Function

#End Region

End Class

**Title:** StudentService.vb

|  |  |
| --- | --- |
| **Description:** | This class is designed to be used with the student form. It is used to perform student related backend tasks such as, getting uncompleted quizzes and save quiz results. |

Imports System.Data.Sql

Imports System.Data.SqlClient

Public Class StudentService

Inherits EntityBase

Public Overrides Sub PromptLogout()

If MessageBox.Show("Are you sure you want to logout? Any unfinished work will be lost", "Prompt", MessageBoxButtons.YesNo) = DialogResult.Yes Then Logout()

End Sub

Public Function GetRandomQuestionsIds(amount As Integer) As List(Of Integer)

OpenConnection()

Dim questionList As New List(Of Integer)

'Selects x amount of random question IDs and returns them.

Dim query As String = String.Format("Select Top {0} QID From tblQuestion Order By NEWID()", amount)

Using SQLCmd As New SqlCommand(query, SQLCon)

Using R As SqlDataReader = SQLCmd.ExecuteReader

While R.Read

questionList.Add(CInt(R("QID")))

End While

End Using

End Using

CloseConnection()

Return questionList

End Function

Public Function GetUncompletedQuizzes(userId As Integer) As List(Of Quiz)

OpenConnection()

Dim quizList As New List(Of Quiz)

' Finds the QuizID and Title of all the quizes the student hasn't completed

Dim query As String = "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 = @0)"

Using SQLCmd As New SqlCommand(query, SQLCon)

SQLCmd.Parameters.AddWithValue("@0", userId)

Using R As SqlDataReader = SQLCmd.ExecuteReader

While R.Read

quizList.Add(New Quiz With {.QuizID = CInt(R("QuizID")), .Title = R("Title").ToString()})

End While

End Using

End Using

CloseConnection()

Return quizList

End Function

Public Function GetQuestionsIds(quizId As Integer) As List(Of Integer)

OpenConnection()

Dim questionIds As New List(Of Integer)

Dim query As String = String.Format("Select QID From tblQuiz Where QuizID = @0 Order By NEWID()")

Using SQLCmd As New SqlCommand(query, SQLCon)

SQLCmd.Parameters.AddWithValue("@0", quizId)

Using R As SqlDataReader = SQLCmd.ExecuteReader

While R.Read

questionIds.Add(CInt(R("QID")))

End While

End Using

End Using

CloseConnection()

Return questionIds

End Function

Public Function AnswerIsCorrect(actualAnswer As String, userAnswer As String) As Boolean

actualAnswer = actualAnswer.Replace(" ", String.Empty).ToLower()

userAnswer = userAnswer.Replace(" ", String.Empty).ToLower()

' Replaces every blank spaces. And to converts all to lower, this makes for more accurate marking

Return userAnswer = actualAnswer

'This will return true if the user answer

'is equal to the actual answer equal, or return false if they are not equal.

End Function

Public Function SaveQuizResult(quizId As Integer, userId As Integer, mark As Integer, totalQuestions As Integer) As Boolean

OpenConnection()

'GetDate() is an inbuilt MSSQL function that gets the current time specific

Dim query As String = "Insert Into tblQuizLog(UserID, QuizID, Mark, Total, DateCompleted) " \_

& "Values(@0, @1, @2, @3, GetDate())"

Using SQLCmd As New SqlCommand(query, SQLCon)

SQLCmd.Parameters.AddWithValue("@0", userId)

SQLCmd.Parameters.AddWithValue("@1", quizId)

SQLCmd.Parameters.AddWithValue("@2", mark)

SQLCmd.Parameters.AddWithValue("@3", totalQuestions)

If SQLCmd.ExecuteNonQuery = 1 Then

Return True

Else

Return False

End If

End Using

End Function

End Class

**Title:** TeacherService.vb

|  |  |
| --- | --- |
| **Description:** | This class is used to perform teacher related backend tasks. For example, it can be used when a teacher wants to find all the student in the teacher class. Get their quiz results. Etc. |

Imports System.Data.SqlClient

Imports System.Text

Public Class TeacherService

Inherits EntityBase

Public Function GetClassInfo(classID As Integer) As String

OpenConnection()

'Gets the Block Name, and Room number of a class ID.

'And returns it in a specific format.

Dim query As String = "Select \* From tblClass Where ClassID = @0"

Using SQLCmd As New SqlCommand(query, SQLCon)

SQLCmd.Parameters.AddWithValue("@0", classID)

Using R As SqlDataReader = SQLCmd.ExecuteReader

While R.Read

Return String.Format("Block: {0} | Room: {1} ", R("Block").ToString(), R("Room").ToString())

End While

End Using

End Using

CloseConnection()

Return String.Empty

End Function

Public Function GetStudents(classID As Integer, name As String) As List(Of User)

OpenConnection()

'This function returns a list of all the students in a teachers class which match

'the name parameter passed down.

Dim studentList As New List(Of User)

'Selects all the students that are in the teacher class.

Dim query As String = "SELECT \* FROM tblUser, tblUserClass" \_

& " WHERE tblUser.Firstname + ' ' + tblUser.Lastname LIKE @0 And tblUser.[Rank] = 0" \_

& " And tblUser.UserID = tblUserClass.UserID And tblUserClass.ClassID = @1 Order By tblUser.Firstname Asc"

Using SQLCmd As New SqlCommand(query, SQLCon)

SQLCmd.Parameters.AddWithValue("@0", "%" & name & "%")

SQLCmd.Parameters.AddWithValue("@1", classID)

Using R As SqlDataReader = SQLCmd.ExecuteReader

If R.HasRows Then

While R.Read

Dim student As New User

student.UserID = CInt(R("UserID"))

student.Username = R("Username").ToString

student.Password = R("Password").ToString

student.Firstname = R("Firstname").ToString

student.Lastname = R("Lastname").ToString

student.Image = CType(R("Picture"), Byte())

studentList.Add(student)

End While

Else

Return Nothing

End If

End Using

End Using

CloseConnection()

Return studentList

End Function

Private Function CreateQuizID() As Integer

'By ordering it by descending, we get the biggest quiz number as the first.

'So then we just increment the biggest number, and we have our new unique QuizID

Dim query As String = "Select Distinct QuizID From tblQuizTitle Order By QuizID Desc"

Dim id As Integer

Try

Using SQLCmd As New SqlCommand(query, SQLCon)

id = CInt(SQLCmd.ExecuteScalar)

End Using

Catch ex As Exception

Return 1

End Try

Return id + 1

End Function

Public Function AddQuiz(questionIdList As List(Of Integer), quizTitle As String) As Boolean

OpenConnection()

'adds al the question Ids in the questionIdList into the database.

Dim query As String = "Insert Into tblQuiz(QuizID, QID) Values(@0, @1)"

Dim quizID As Integer = CreateQuizID()

For Each questionId As Integer In questionIdList

Using SQLCmd As New SqlCommand(query, SQLCon)

SQLCmd.Parameters.AddWithValue("@0", quizID)

SQLCmd.Parameters.AddWithValue("@1", questionId)

SQLCmd.ExecuteNonQuery()

End Using

Next

'Adds the QuizTitle add the end on a seperate table.

Return AddQuizTitle(quizID, quizTitle)

End Function

Private Function AddQuizTitle(quizId As Integer, title As String) As Boolean

Dim query As String = "Insert Into tblQuizTitle(QuizID, Title) Values(@0, @1)"

Using SQLCmd As New SqlCommand(query, SQLCon)

'We paramatize just incase the user enters something that could tamper with the system database.

SQLCmd.Parameters.AddWithValue("@0", quizId)

SQLCmd.Parameters.AddWithValue("@1", title)

If SQLCmd.ExecuteNonQuery() = 1 Then

Return True

Else

Return False

End If

End Using

End Function

Public Function GetStudentQuizResults(userId As Integer, dateFrom As DateTime, dateTo As DateTime) As List(Of Quiz)

OpenConnection()

' Selects all the completed quizzes in order of the latest one first for the student.

Dim query As String = "Select \* From tblQuizTitle, tblQuizLog " \_

& "Where tblQuizTitle.QuizID = tblQuizLog.QuizID And UserID = @userId " \_

& "And DateCompleted Between @from And @to Order By DateCompleted Asc"

Dim quizList As New List(Of Quiz)

'a list where we add all the Quizzes found.

Using SQLCmd As New SqlCommand(query, SQLCon)

SQLCmd.Parameters.AddWithValue("@from", dateFrom)

SQLCmd.Parameters.AddWithValue("@to", dateTo)

SQLCmd.Parameters.AddWithValue("@userId", userId)

Using R As SqlDataReader = SQLCmd.ExecuteReader

If R.HasRows Then

While R.Read

Dim percent As Integer = CalculatePercentage(CInt(R("Total")), CInt(R("Mark")))

quizList.Add(New Quiz With {.Percentage = percent, .Title = R("Title").ToString()})

End While

'Adds each found quiz into the quizList

Else

Return Nothing

End If

End Using

End Using

CloseConnection()

'Returns the list

Return quizList

End Function

Private Function CalculatePercentage(total As Integer, mark As Integer) As Integer

Return CInt((mark / total) \* 100)

End Function

Public Sub FetchQuiz()

OpenConnection()

Dim quizList As New List(Of Quiz)

'Selects all the Unique QuizIDs of the Quizzes that are in the database.

Dim query As String = "Select Distinct tblQuiz.QuizID, tblQuizTitle.Title From tblQuiz, tblQuizTitle " \_

& "Where tblQuiz.QuizID = tblQuizTitle.QuizID"

Using SQLCmd As New SqlCommand(query, SQLCon)

Using R As SqlDataReader = SQLCmd.ExecuteReader

While R.Read

quizList.Add(New Quiz With {.QuizID = CInt(R("QuizID")), .Title = R("Title").ToString})

End While

End Using

End Using

AddQuizToTree(quizList)

CloseConnection()

End Sub

Public Sub PopulateTreeView(grade As String, topic As String)

OpenConnection()

Dim query As String = "Select \* From tblQuestion Where Grade LIKE @0 And Topic LIKE @1"

Using SQLCmd = New SqlCommand(query, SQLCon)

SQLCmd.Parameters.AddWithValue("@0", "%" & grade & "%")

SQLCmd.Parameters.AddWithValue("@1", "%" & topic & "%")

Using R As SqlDataReader = SQLCmd.ExecuteReader

If R.HasRows Then

Dim parentNode As TreeNode = frmTeacher.tvQuestions.Nodes.Add(topic) 'e.g Algebra

'Adds each question fetched under the parent node.

'So under the ParentNode Algebra. All the algebra questions will be added under it.

While R.Read

parentNode.Nodes.Add(New CustomTreeNode() With {.QuestionID = CInt(R("QID")), .Text = R("Question").ToString})

End While

End If

End Using

End Using

CloseConnection()

End Sub

Private Sub AddQuizToTree(quizList As List(Of Quiz))

For Each quiz As Quiz In quizList

'Selects all the Questions in a quiz

Dim query As String = "Select \* From tblQuiz, tblQuestion " \_

& "Where tblQuiz.QID = tblQuestion.QID And QuizID = @0"

Dim parentNode As TreeNode = frmTeacher.tvQuestions.Nodes.Add(quiz.Title)

Using SQLCmd As New SqlCommand(query, SQLCon)

SQLCmd.Parameters.AddWithValue("@0", quiz.QuizID)

Using R As SqlDataReader = SQLCmd.ExecuteReader()

While R.Read

parentNode.Nodes.Add(New CustomTreeNode() With {.QuestionID = CInt(R("QID")), .Text = R("Question").ToString})

End While

'Adds them to the node of the TreeView in frmTeacher.

End Using

End Using

Next

End Sub

Public Function ExportQuestions(quesionIdList As List(Of Integer)) As String

OpenConnection()

Dim sb As New StringBuilder

Dim counter As Integer = 1

'Selects the question and answer of each question in the questionIdList

For Each qID As Integer In quesionIdList

Dim query As String = "Select Question, Answer From tblQuestion Where QID = @0"

Using SQLCmd As New SqlCommand(query, SQLCon)

SQLCmd.Parameters.AddWithValue("@0", qID)

Using R As SqlDataReader = SQLCmd.ExecuteReader()

While R.Read

sb.AppendLine(String.Format("{0}) Question: {1}", counter, R("Question").ToString()))

sb.AppendLine(String.Format("Answer: {0}", R("Answer").ToString()))

sb.AppendLine()

End While

End Using

End Using

counter += 1

Next

'Builds a string of the question and answers. Then returns it.

CloseConnection()

Return sb.ToString()

End Function

Private Function GetStudentQuizInfo(studentId As Integer) As StudentQuizInfo

Dim student As New StudentQuizInfo With {.UserID = studentId}

'This structure has the name of the student, and an array of the percenatage

'they got on each quiz.

student.QuizScores = New List(Of Integer)

'Selects all the quizzes a user has completed

Dim query As String = "Select \* From tblUser, tblQuizLog " \_

& "Where tblUser.UserID = tblQuizLog.UserID " \_

& "And tblUser.UserID = @0"

Using SQLCmd As New SqlCommand(query, SQLCon)

SQLCmd.Parameters.AddWithValue("@0", studentId)

Using R As SqlDataReader = SQLCmd.ExecuteReader

While R.Read

student.Firstname = R("Firstname").ToString()

student.Lastname = R("Lastname").ToString().ToString()

Dim percent As Integer = CalculatePercentage(CInt(R("Total")), CInt(R("Mark")))

student.QuizScores.Add(percent)

End While

End Using

For Each score As Integer In student.QuizScores

student.Average += score

'Fills in the avrage score the student got by adding up each item

'In the quizScores array.

Next

student.Average = CInt(student.Average / student.QuizScores.Count)

'Calculates the average of their score

Return student

End Using

End Function

Public Function GetStudentsQuizResults(classId As Integer) As List(Of StudentQuizInfo)

OpenConnection()

'This query gets a distinct list of the studentIds of all the students who have

'completed a quiz from a specific class

Dim query As String = "SELECT Distinct tblUser.UserID FROM tblUser, tblQuizLog, tblUserClass " \_

& "Where tblQuizLog.UserID = tblUser.UserID " \_

& "And tblUserClass.UserID = tblUser.UserID " \_

& "And tblUserClass.ClassID = @0"

Dim studentIDs As New List(Of Integer)

Using SQLCmd As New SqlCommand(query, SQLCon)

SQLCmd.Parameters.AddWithValue("@0", classId)

Using R As SqlDataReader = SQLCmd.ExecuteReader

While R.Read

studentIDs.Add(CInt(R("UserID")))

'Adds of the UserIDs of all the students found into a list/array

End While

End Using

End Using

Dim studentsList As New List(Of StudentQuizInfo)

'This will be an array/list of students. And for each student

'it contains the percentage they got on each quiz they have ever taken.

For Each id As Integer In studentIDs

studentsList.Add(GetStudentQuizInfo(id))

Next

'We then sort the list. We sort by the average which is all the quizzes percentages added together

'divided by the total amount of quizzes taken.

Dim sortedList As List(Of StudentQuizInfo) = Quicksort.Sort(studentsList)

CloseConnection()

Return sortedList

End Function

End Class

**Title:** QuickSort

|  |  |
| --- | --- |
| **Description:** | This class is a quick sort class which sorts a List of the StudentQuizInfo object/structure. This class is inside of the TeacherService class. |

Private Class Quicksort

Public Shared Function Sort(arrayList As List(Of StudentQuizInfo)) As List(Of StudentQuizInfo)

'If list is empty of has 1 item then its already sorted

If arrayList.Count <= 1 Then

Return arrayList

Else

'Makes a call to the recursive sort algorithm

Dim sortedList As List(Of StudentQuizInfo) = SortList(0, arrayList.Count - 1, arrayList)

Return sortedList

End If

End Function

Private Shared Function SortList(lowIndex As Integer, hiIndex As Integer, arrayList As List(Of StudentQuizInfo)) As List(Of StudentQuizInfo)

If hiIndex > lowIndex Then

Dim initialPivot As Integer = GetPivotPosition(lowIndex, hiIndex, arrayList)

SortList(lowIndex, initialPivot - 1, arrayList)

SortList(initialPivot + 1, hiIndex, arrayList)

End If

Return arrayList 'Returns the sorted list at the end of the recusive loop

End Function

Private Shared Function GetPivotPosition(lowIndex As Integer, hiIndex As Integer, arrayList As List(Of StudentQuizInfo)) As Integer

Dim x As Integer = lowIndex

'Uses the first index as the starting point

Dim y As Integer = hiIndex - 1

'The pivot is going to be used to compare each item in the list

Dim pivot As Integer = arrayList(hiIndex).Average

Do

'Compares each item in the left sub-list that is greater than the pivot

While arrayList(x).Average <= pivot AndAlso x < hiIndex

x += 1

End While

'Compares each item in the right sub-list that is less than the pivot

While arrayList(y).Average >= pivot AndAlso y > lowIndex

y -= 1

End While

If x < y Then Swap(x, y, arrayList) ' Swaps X and Y

Loop While x < y 'Keeps looping until Y is greater than X

' Swaps the data from the right to the new position in the list

If arrayList(x).Average > pivot Then Swap(x, hiIndex, arrayList) 'Exchanges the data

Return x ' Returns the position of the pivot

End Function

Private Shared Sub Swap(indexFrom As Integer, indexTo As Integer, ByRef arrayList As List(Of StudentQuizInfo))

Dim selectedStudent As StudentQuizInfo = arrayList(indexFrom) 'selected student to swap

arrayList(indexFrom) = arrayList(indexTo)

arrayList(indexTo) = selectedStudent

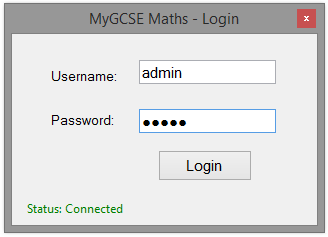
'Swaps the current with the specified index

End Sub

End Class

**Title:** frmLogin.vb

|  |  |
| --- | --- |
| **Description:** | This is the login form. This is used when the user is trying to login to the system. |



Imports MyGCSE\_Maths.EntityBase

Imports System.Threading

Public Class frmLogin

Private Login As New LoginService

#Region "Global controls declerations"

Friend WithEvents txtUsername As New TextBox

Friend WithEvents txtPassword As New TextBox

Friend WithEvents btnLogin As New Button

Friend WithEvents lblConStatus As New Label

Private Sub CreateControls()

Dim lblUsername As New Label

lblUsername.AutoSize = True

lblUsername.Font = New Font("Microsoft Sans Serif", 9.75!, FontStyle.Regular)

lblUsername.Location = New Point(36, 34)

lblUsername.Text = "Username:"

txtUsername.Font = New Font("Microsoft Sans Serif", 11.25!, FontStyle.Regular)

txtUsername.Location = New Point(127, 26)

txtUsername.Size = New Size(137, 24)

txtUsername.TabIndex = 1

Dim lblPassword As New Label

lblPassword.AutoSize = True

lblPassword.Font = New Font("Microsoft Sans Serif", 9.75!, FontStyle.Regular)

lblPassword.Location = New Point(36, 78)

lblPassword.Text = "Password:"

txtPassword.Font = New Font("Microsoft Sans Serif", 11.25!, FontStyle.Regular)

txtPassword.Location = New Point(127, 75)

txtPassword.Size = New Size(137, 24)

txtPassword.TabIndex = 2

txtPassword.UseSystemPasswordChar = True

btnLogin.Font = New Font("Microsoft Sans Serif", 11.25!, FontStyle.Regular)

btnLogin.Location = New Point(146, 116)

btnLogin.Size = New Size(94, 31)

btnLogin.TabIndex = 3

btnLogin.Text = "Login"

lblConStatus.AutoSize = True

lblConStatus.Font = New Font("Segoe UI", 9.0!, FontStyle.Regular)

lblConStatus.Location = New Point(12, 167)

lblConStatus.Text = "Connecting..."

Dim ctrlCollection As New Collection From {btnLogin, txtUsername, txtPassword, lblUsername, lblPassword, lblConStatus}

For Each ctrl As Control In ctrlCollection

Me.Controls.Add(ctrl)

Next

End Sub

Sub New()

' This call is required by the designer.

InitializeComponent()

' Add any initialization after the InitializeComponent() call.

CreateControls()

End Sub

#End Region

Private Sub txtPassword\_KeyDown(sender As Object, e As KeyEventArgs) Handles txtPassword.KeyDown

If e.KeyCode = Keys.Enter Then

e.SuppressKeyPress = True

btnLogin.PerformClick()

End If

End Sub

Private Sub LoginForm\_Shown(sender As Object, e As EventArgs) Handles MyBase.Shown

btnLogin.Enabled = False

Dim t As New Thread(Sub() CheckForConnection())

t.Start()

'ThreadPool.QueueUserWorkItem(Sub() CheckForConnection())

End Sub

Private Sub CheckForConnection()

If Login.HasConnection() Then

UpdateLabel("Status: Connected", Color.Green)

Else

UpdateLabel("Status: Connection failed", Color.Red)

End If

Invoke(Sub() btnLogin.Enabled = True)

End Sub

Private Sub UpdateLabel(text As String, foreColor As Color)

' Because we're trying to change the properties of a controls that's on a seperate thread, we need to make

' a safe call to the control, we do that by invoking the method using a delegate sub in order for it to be handled correctly.

' Unless we 'd get a CrossThreadCall Exception thrown

Invoke(Sub() lblConStatus.Text = text)

Invoke(Sub() lblConStatus.ForeColor = foreColor)

End Sub

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

If Login.IsConnected Then

If Login.CorrectLogin(txtUsername.Text, txtPassword.Text) Then

ShowForm(MyUser.Rank)

Else

MessageBox.Show("Incorrect username or password", "Alert")

End If

Else

MessageBox.Show("Cannot connect to server", "Noice")

End If

End Sub

Private Sub ShowForm(userRank As Rank)

' After a successful login, this sub is called. It will show the user the correct UI depending on their rank

Select Case userRank

Case Rank.Admin

frmAdmin.Show()

Case Rank.Teacher

frmTeacher.Show()

Case Rank.Student

frmStudent.Show()

End Select

Me.Hide()

txtUsername.Clear()

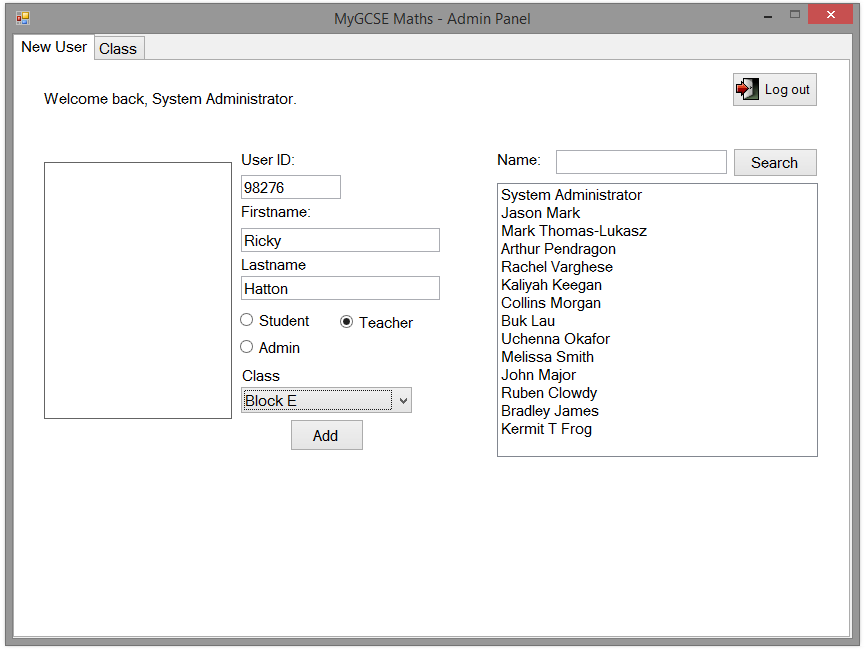
txtPassword.Clear()

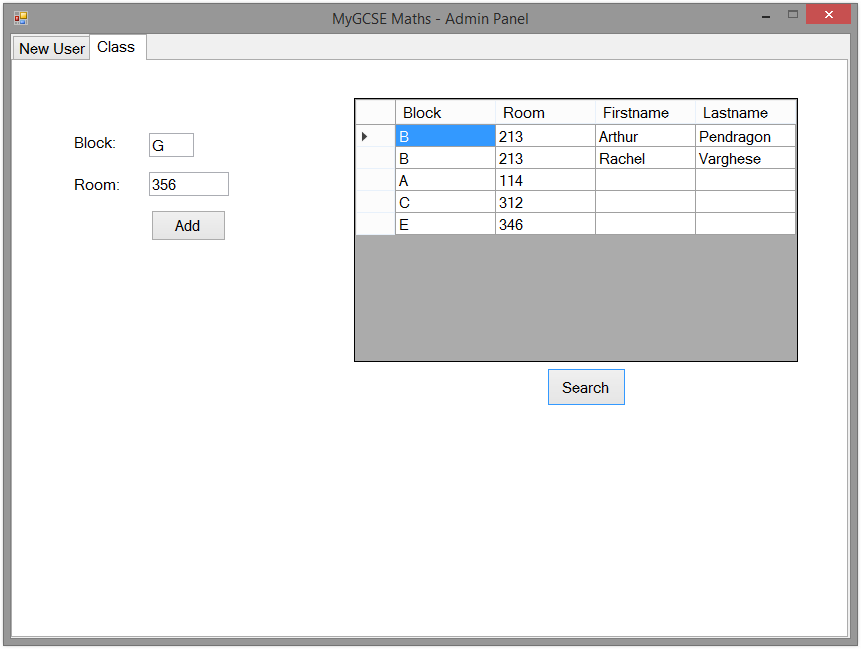
End Sub

End Class

**Title:** frmAdmin.vb

|  |  |
| --- | --- |
| **Description:** | This is the admin panel. This is what the admin uses to mamage user accounts for the system. They can create classes, add teachers & students to classes and see what class each teacher teaches. Also the admin can manipulate the user account. For example, they can delete, editing user accounts too. |



Imports MyGCSE\_Maths.EntityBase

Public Class frmAdmin

Private Admin As New AdminService

#Region "Glocal Control Declrations"

Friend WithEvents btnLogout As New Button

Friend WithEvents lblWelcomeBack As New Label

Friend WithEvents picUser As New PictureBox

Friend WithEvents txtUserID As New TextBox

Friend WithEvents txtLastname As New TextBox

Friend WithEvents txtFirstname As New TextBox

Friend WithEvents rdoStudent As New RadioButton

Friend WithEvents rdoTeacher As New RadioButton

Friend WithEvents rdoAdmin As New RadioButton

Friend WithEvents btnAdd As New Button

Friend WithEvents lblClassInfo As New Label

Friend WithEvents cboClass As New ComboBox

Friend WithEvents lstUsers As New ListBox

Friend WithEvents txtSearch As New TextBox

Friend WithEvents btnSearch As New Button

Private Sub CreateTab1Controls()

lblWelcomeBack.AutoSize = True

lblWelcomeBack.Location = New Point(24, 28)

lblWelcomeBack.Text = "Welcome back, "

btnLogout.Font = New Font("Microsoft Sans Serif", 9.75!, FontStyle.Regular)

btnLogout.Image = My.Resources.logout\_img

btnLogout.ImageAlign = ContentAlignment.MiddleLeft

btnLogout.TextAlign = ContentAlignment.MiddleRight

btnLogout.Location = New Point(715, 11)

btnLogout.Size = New Size(86, 35)

btnLogout.Text = "Log out"

picUser.BorderStyle = BorderStyle.FixedSingle

picUser.Cursor = Cursors.Hand

picUser.Location = New Point(27, 101)

picUser.Size = New Size(188, 257)

picUser.SizeMode = PictureBoxSizeMode.StretchImage

Dim lblUserID As New Label

lblUserID.AutoSize = True

lblUserID.Location = New Point(221, 89)

lblUserID.Text = "User ID: "

txtUserID.Location = New Point(224, 114)

txtUserID.Size = New Size(100, 24)

btnSearch.Location = New Point(716, 87)

btnSearch.Size = New Size(85, 29)

btnSearch.Text = "Search"

rdoStudent.AutoSize = True

rdoStudent.Location = New Point(223, 248)

rdoStudent.Text = "Student"

rdoTeacher.AutoSize = True

rdoTeacher.Location = New Point(323, 250)

rdoTeacher.Text = "Teacher"

rdoAdmin.AutoSize = True

rdoAdmin.Location = New Point(223, 275)

rdoAdmin.Text = "Admin"

Dim lblLastname As New Label

lblLastname.AutoSize = True

lblLastname.Location = New Point(221, 194)

lblLastname.Text = "Lastname"

Dim lblFirstname As New Label

lblFirstname.AutoSize = True

lblFirstname.Location = New Point(221, 141)

lblFirstname.Text = "Firstname:"

txtLastname.Location = New Point(224, 215)

txtLastname.Size = New Size(199, 24)

txtFirstname.Location = New Point(224, 167)

txtFirstname.Size = New Size(199, 24)

lblClassInfo.AutoSize = True

lblClassInfo.Location = New Point(222, 305)

lblClassInfo.Text = "Class"

lblClassInfo.Visible = False

cboClass.DropDownStyle = ComboBoxStyle.DropDownList

cboClass.Location = New Point(224, 326)

cboClass.Size = New Size(171, 26)

cboClass.Visible = False

btnAdd.Enabled = False

btnAdd.Location = New Point(273, 358)

btnAdd.Size = New Size(74, 32)

btnAdd.Text = "Add"

Dim lblName As New Label

lblName.AutoSize = True

lblName.Location = New Point(477, 89)

lblName.Text = "Name: "

txtSearch.Location = New Point(539, 89)

txtSearch.Size = New Size(171, 24)

lstUsers.Location = New Point(480, 122)

lstUsers.Size = New Size(321, 274)

Dim ctrlCollection As New Collection From {lblWelcomeBack, btnLogout, lstUsers, txtSearch, lblName, btnSearch, rdoStudent, rdoTeacher, rdoAdmin, picUser, lblLastname, lblUserID, txtUserID, lblFirstname, txtLastname, txtFirstname, btnAdd, cboClass, lblClassInfo}

For Each ctrl As Control In ctrlCollection

Me.tabAdd.Controls.Add(ctrl)

Next

End Sub

Friend WithEvents txtBlock As New TextBox

Friend WithEvents txtRoom As New TextBox

Friend WithEvents btnSearchTeacher As New Button

Friend WithEvents btnAddClass As New Button

Friend WithEvents dgvClassrooms As New DataGridView

Private Sub CreateTab2Controls()

Dim lblRoom As New Label

lblRoom.AutoSize = True

lblRoom.Location = New Point(56, 114)

lblRoom.Text = "Room:"

txtRoom.Location = New Point(134, 111)

txtRoom.Size = New Size(80, 24)

Dim lblBlock As New Label

lblBlock.AutoSize = True

lblBlock.Location = New Point(56, 72)

lblBlock.Text = "Block:"

txtBlock.Location = New Point(134, 72)

txtBlock.MaxLength = 1

txtBlock.Size = New Size(45, 24)

btnAddClass.Location = New Point(136, 149)

btnAddClass.Size = New Size(75, 31)

btnAddClass.Text = "Add"

dgvClassrooms.ReadOnly = True

dgvClassrooms.AllowUserToAddRows = False

dgvClassrooms.ColumnHeadersHeightSizeMode = DataGridViewColumnHeadersHeightSizeMode.AutoSize

Dim columnBlock As New DataGridViewTextBoxColumn With {.HeaderText = "Block"}

Dim columnRoom As New DataGridViewTextBoxColumn With {.HeaderText = "Room"}

Dim columnFirstname As New DataGridViewTextBoxColumn With {.HeaderText = "Firstname"}

Dim columnLastname As New DataGridViewTextBoxColumn With {.HeaderText = "Lastname"}

dgvClassrooms.Columns.AddRange({columnBlock, columnRoom, columnFirstname, columnLastname})

dgvClassrooms.Location = New Point(339, 37)

dgvClassrooms.Size = New Size(444, 264)

btnSearchTeacher.Location = New Point(532, 307)

btnSearchTeacher.Size = New Size(79, 38)

btnSearchTeacher.Text = "Search"

Dim ctrlCollection As New Collection From {lblBlock, lblRoom, txtBlock, txtRoom, btnSearchTeacher, btnAddClass, dgvClassrooms}

For Each ctrl As Control In ctrlCollection

Me.tabClass.Controls.Add(ctrl)

Next

End Sub

Sub New()

' This call is required by the designer.

InitializeComponent()

' Add any initialization after the InitializeComponent() call.

CreateTab1Controls()

CreateTab2Controls()

End Sub

#End Region

Private Sub AdminForm\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

lblWelcomeBack.Text = String.Format("Welcome back, {0} {1}.", MyUser.Firstname, MyUser.Lastname)

rdoStudent.Checked = True

End Sub

Private Sub frmAdmin\_Closing(sender As Object, e As EventArgs) Handles MyBase.FormClosed

Application.Exit()

End Sub

Private Function GetRank() As Rank

If rdoAdmin.Checked Then

Return Rank.Admin

ElseIf rdoTeacher.Checked Then

Return Rank.Teacher

ElseIf rdoStudent.Checked Then

Return Rank.Student

Else

Return Nothing

'Implying that one of the radio buttons must be checked unless this function can't be executed

'Putting this so the compiler will stop giving warnings.

End If

End Function

Private Function GetClassID() As Integer

'We cast the selectedItem object to the ClassRoom object, then access the class Id

'We then return the ClassID extracted.

If GetRank() = Rank.Student Or GetRank() = Rank.Teacher Then

Return DirectCast(cboClass.SelectedItem, ClassRoom).ClassID

Else

Return 0

End If

End Function

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

'I didn't validate the txtFirstname or txtLastname to check if any of th textboxes are empty in this subroutine.

'I used another sub routine which will enable the add button if the text boxes are filled in.

'Therefore this btnClick.Click event won't get raised unless it meets the criteria of ActivateAddBtn() sub.

If Integer.TryParse(txtUserID.Text, Nothing) Then

If MessageBox.Show("Are you sure you want to add this user?", "Prompt", MessageBoxButtons.YesNo) = DialogResult.Yes Then

Try

If Admin.AddAccount(CInt(txtUserID.Text), txtFirstname.Text, txtLastname.Text, GetRank(), picUser.Image.ToBytes(), GetClassID()) Then

MessageBox.Show("User added successfully", "Notice")

ClearControls()

RefreshClassList()

Else

MessageBox.Show("An error occured whilst to add a new user")

End If

Catch ex As Exception

MessageBox.Show(ex.Message)

End Try

End If

Else

MessageBox.Show("User rank must be selected, and UserID must only consist of numbers")

End If

End Sub

Private Sub ClearControls()

' Clears or resets items in the controls

txtFirstname.Clear()

txtLastname.Clear()

txtUserID.Clear()

cboClass.SelectedIndex = -1

picUser.Image = Nothing

End Sub

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

lstUsers.Items.Clear()

'Gets a list of all users whos name match the text in txtSearch

Dim usersList As List(Of User) = Admin.GetUsers(txtSearch.Text)

'If the GetUser() function returns nothing it means no user was found.

If IsNothing(usersList) Then

MessageBox.Show("User not found")

Else

For Each user As User In usersList

lstUsers.Items.Add(user)

Next

End If

End Sub

Private Sub lstUsers\_SelectedIndexChanged(sender As Object, e As EventArgs) Handles lstUsers.SelectedIndexChanged

'This is when a user from the listbox is selected.

If lstUsers.SelectedIndex >= 0 Then

RefreshClassList() 'Refreshes the class list

'Because we insert the object User to the ListBox.Items, it means that each SelectedItem

'Is of User. But the SelectedItem property is an Object. So we need to convert the SelectedItem

'Property to the User Object. Then we can directly access the Object as a User object.

Dim selectedUser As User = DirectCast(lstUsers.SelectedItem, User)

GetUserInfo(selectedUser.UserID, selectedUser.Rank)

frmEdit.Show()

frmEdit.BringToFront()

End If

End Sub

Private Sub GetUserInfo(userId As Integer, userRank As Rank)

Dim selectedUser As User = Admin.GetUserInformation(userId, userRank)

frmEdit.txtUserID.Text = selectedUser.UserID.ToString()

frmEdit.txtUsername.Text = selectedUser.Username

frmEdit.txtPassword.Text = selectedUser.Password

frmEdit.cboRank.SelectedIndex = CInt(selectedUser.Rank)

If selectedUser.Rank <> Rank.Admin Then frmEdit.cboClass.Text = selectedUser.ClassName

frmEdit.txtFirstname.Text = selectedUser.Firstname

frmEdit.txtLastname.Text = selectedUser.Lastname

frmEdit.txtEmail.Text = selectedUser.Email

frmEdit.picUser.Image = Image.FromStream(selectedUser.Image.ToStream())

End Sub

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

Try

Using ofd As New OpenFileDialog With {.Filter = "Images|\*.jpg;\*.png;\*.bmp"}

If ofd.ShowDialog = DialogResult.OK Then picUser.Image = Image.FromFile(ofd.FileName)

End Using

Catch ex As Exception

MessageBox.Show(ex.Message)

End Try

End Sub

Private Sub txtSearch\_KeyDown(sender As Object, e As KeyEventArgs) Handles txtSearch.KeyDown

If e.KeyCode = Keys.Enter Then

e.SuppressKeyPress = True

btnSearch.PerformClick()

End If

End Sub

Private Sub rdoStudent\_CheckedChanged(sender As Object, e As EventArgs) Handles rdoStudent.CheckedChanged, rdoAdmin.CheckedChanged, rdoTeacher.CheckedChanged

If rdoStudent.Checked Or rdoTeacher.Checked Then

RefreshClassList()

LblsVisible(True)

Else

LblsVisible(False)

End If

End Sub

Private Sub LblsVisible(value As Boolean)

lblClassInfo.Visible = value

cboClass.Visible = value

End Sub

Private Sub RefreshClassList()

'Refreshes the list of all the classes in the system

'And adds them to the combo box

frmEdit.cboClass.Items.Clear()

Me.cboClass.Items.Clear()

Dim classList = Admin.GetAllClasses

For Each item As ClassRoom In classList

frmEdit.cboClass.Items.Add(item)

Me.cboClass.Items.Add(item)

Next

End Sub

Private Sub ActivateAddBtn() Handles txtLastname.TextChanged, cboClass.SelectedIndexChanged, \_

rdoAdmin.CheckedChanged, rdoTeacher.CheckedChanged, rdoStudent.CheckedChanged

btnAdd.Enabled = False

If Not String.IsNullOrWhiteSpace(txtLastname.Text) Or Not String.IsNullOrWhiteSpace(txtFirstname.Text) Or Not String.IsNullOrWhiteSpace(txtUserID.Text) Then

If rdoStudent.Checked Or rdoTeacher.Checked Then

If cboClass.SelectedIndex >= 0 Then btnAdd.Enabled = True

Else

btnAdd.Enabled = True

End If

End If

'This subroutine checks if all the user has filled in all the infroamtion correctly.

'This will only enable the add button if all the conditions are met

End Sub

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

Admin.PromptLogout()

End Sub

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

'Checks to see if any of the textboxe are empty before adding.

If String.IsNullOrWhiteSpace(txtBlock.Text) = False And String.IsNullOrWhiteSpace(txtRoom.Text) = False Then

If MessageBox.Show("Are you sure you want to add this class?", "Notice", MessageBoxButtons.YesNo) = DialogResult.Yes Then

If Admin.CreateClass(CChar(txtBlock.Text), txtRoom.Text) Then

MessageBox.Show("Class created successfully!")

Else

MessageBox.Show("An error occured whilst adding class")

End If

End If

Else

MessageBox.Show("One or more fields have not been filled in", "Alert")

End If

End Sub

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

dgvClassrooms.Rows.Clear()

'Calls a subroutine which will check for all the classes in the system and add them to the datagrid

'It also adds all the empty classes, and all the teachers that are assigned to a class.

Admin.AddOccupiedClasses()

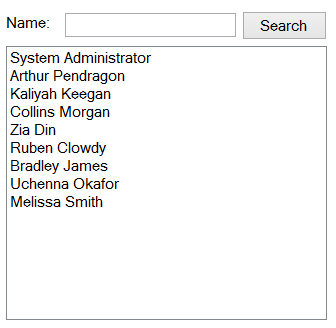
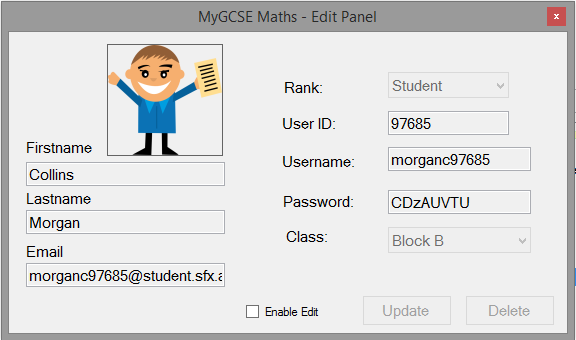
Admin.AddEmptyClasses()

End Sub

End Class

**Title:** frmEdit.vb

|  |  |
| --- | --- |
| **Description:** | This form is used to edit or to delete a users account. This form is called from the Admin Form (frmAdmin). |



Imports System.Text.RegularExpressions

Imports MyGCSE\_Maths.EntityBase

Public Class frmEdit

Private Admin As New AdminService

#Region "Global control declerations"

Friend WithEvents btnDelete As New Button

Friend WithEvents btnUpdate As New Button

Friend WithEvents chkBoxEdit As New CheckBox

Friend WithEvents lblClass As New Label

Friend WithEvents cboClass As New ComboBox

Friend WithEvents lblRank As New Label

Friend WithEvents cboRank As New ComboBox

Friend WithEvents txtPassword As New TextBox

Friend WithEvents txtUserID As New TextBox

Friend WithEvents txtUsername As New TextBox

Friend WithEvents picUser As New PictureBox

Friend WithEvents txtEmail As New TextBox

Friend WithEvents txtLastname As New TextBox

Friend WithEvents txtFirstname As New TextBox

Sub New()

' This call is required by the designer.

InitializeComponent()

' Add any initialization after the InitializeComponent() call.

CreateControls()

End Sub

Private Sub CreateControls()

'MyGCSE Maths reigion

lblRank.AutoSize = True

lblRank.Font = New Font("Microsoft Sans Serif", 11.25!, FontStyle.Regular)

lblRank.Location = New Point(272, 46)

lblRank.Text = "Rank:"

cboRank.DropDownStyle = ComboBoxStyle.DropDownList

cboRank.Enabled = False

cboRank.Font = New Font("Microsoft Sans Serif", 11.25!, FontStyle.Regular)

cboRank.Items.AddRange({"Student", "Teacher", "Admin"})

cboRank.Location = New Point(379, 40)

cboRank.Size = New Size(121, 26)

Dim lblUserID As New Label

lblUserID.AutoSize = True

lblUserID.Font = New Font("Microsoft Sans Serif", 11.25!, FontStyle.Regular)

lblUserID.Location = New Point(270, 82)

lblUserID.Text = "User ID:"

txtUserID.Font = New Font("Microsoft Sans Serif", 11.25!, FontStyle.Regular)

txtUserID.Location = New Point(379, 79)

txtUserID.Size = New Size(121, 24)

Dim lblUsername As New Label

lblUsername.AutoSize = True

lblUsername.Font = New Font("Microsoft Sans Serif", 11.25!, FontStyle.Regular)

lblUsername.Location = New Point(270, 120)

lblUsername.Text = "Username:"

txtUsername.Font = New Font("Microsoft Sans Serif", 11.25!, FontStyle.Regular)

txtUsername.Location = New Point(379, 115)

txtUsername.Size = New Size(143, 24)

Dim lblPassword As New Label

lblPassword.AutoSize = True

lblPassword.Font = New Font("Microsoft Sans Serif", 11.25!, FontStyle.Regular)

lblPassword.Location = New Point(271, 160)

lblPassword.Text = "Password:"

txtPassword.Font = New Font("Microsoft Sans Serif", 11.25!, FontStyle.Regular)

txtPassword.Location = New Point(379, 158)

txtPassword.Size = New Size(143, 24)

Dim lblClass As New Label

lblClass.AutoSize = True

lblClass.Font = New Font("Microsoft Sans Serif", 11.25!, FontStyle.Regular)

lblClass.Location = New Point(274, 195)

lblClass.Text = "Class:"

cboClass.DropDownStyle = ComboBoxStyle.DropDownList

cboClass.Enabled = False

cboClass.Font = New Font("Microsoft Sans Serif", 11.25!, FontStyle.Regular)

cboClass.Location = New Point(379, 195)

cboClass.Size = New Size(143, 26)

chkBoxEdit.AutoSize = True

chkBoxEdit.Location = New Point(237, 272)

chkBoxEdit.Text = "Enable Edit"

btnDelete.Enabled = False

btnDelete.Font = New Font("Microsoft Sans Serif", 11.25!, FontStyle.Regular)

btnDelete.Location = New Point(456, 263)

btnDelete.Size = New Size(90, 31)

btnDelete.Text = "Delete"

btnUpdate.Enabled = False

btnUpdate.Font = New Font("Microsoft Sans Serif", 11.25!, FontStyle.Regular)

btnUpdate.Location = New Point(353, 263)

btnUpdate.Size = New Size(90, 31)

btnUpdate.Text = "Update"

'User Information region

picUser.BorderStyle = BorderStyle.FixedSingle

picUser.Cursor = Cursors.Hand

picUser.Enabled = False

picUser.Location = New Point(98, 12)

picUser.Size = New Size(116, 112)

picUser.SizeMode = PictureBoxSizeMode.StretchImage

Dim lblFirstname As New Label

lblFirstname.AutoSize = True

lblFirstname.Font = New Font("Microsoft Sans Serif", 11.25!, FontStyle.Regular)

lblFirstname.Location = New Point(14, 106)

lblFirstname.Text = "Firstname"

txtFirstname.Font = New Font("Microsoft Sans Serif", 11.25!, FontStyle.Regular)

txtFirstname.Location = New Point(17, 130)

txtFirstname.Size = New Size(199, 24)

Dim lblLastname As New Label

lblLastname.AutoSize = True

lblLastname.Font = New Font("Microsoft Sans Serif", 11.25!, FontStyle.Regular)

lblLastname.Location = New Point(14, 157)

lblLastname.Text = "Lastname"

txtLastname.Font = New Font("Microsoft Sans Serif", 11.25!, FontStyle.Regular)

txtLastname.Location = New Point(17, 178)

txtLastname.Size = New Size(199, 24)

Dim lblEmail As New Label

lblEmail.AutoSize = True

lblEmail.Font = New Font("Microsoft Sans Serif", 11.25!, FontStyle.Regular)

lblEmail.Location = New Point(14, 210)

lblEmail.Text = "Email"

txtEmail.Font = New Font("Microsoft Sans Serif", 11.25!, FontStyle.Regular)

txtEmail.Location = New Point(17, 231)

txtEmail.Size = New Size(199, 24)

Dim ctrlCollection As New Collection From {chkBoxEdit, btnDelete, btnUpdate, lblClass, lblRank, cboRank, cboClass, lblPassword, lblUserID, lblUsername, txtPassword, txtUserID, txtUsername, picUser, lblEmail, lblLastname, lblFirstname, txtEmail, txtLastname, txtFirstname}

For Each ctrl As Control In ctrlCollection

Me.Controls.Add(ctrl)

Next

For Each txtbx As TextBox In Me.Controls.OfType(Of TextBox)()

txtbx.ReadOnly = True

Next

End Sub

#End Region

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

'If the check box is checked. It will enable all the controls

'If its unchecked it will disable all the controls.

'This is to prevent mistakes from being made when editing user information.

If chkBoxEdit.Checked Then

ChangeState(True)

Else

ChangeState(False)

End If

End Sub

Private Sub ChangeState(value As Boolean)

cboClass.Enabled = value

btnDelete.Enabled = value

btnUpdate.Enabled = value

picUser.Enabled = value

'Not Value == Opposite of value.

txtPassword.ReadOnly = Not value

txtFirstname.ReadOnly = Not value

txtLastname.ReadOnly = Not value

txtEmail.ReadOnly = Not value

End Sub

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

If MessageBox.Show("Are you sure you want to delete this record?", "Prompt", MessageBoxButtons.YesNo) = DialogResult.Yes Then

If Admin.DeleteAccount(CInt(txtUserID.Text)) Then

MessageBox.Show("User deleted successfully!", "Notice")

Me.Close()

frmAdmin.btnSearch.PerformClick() 'Clicks the refresh button, this in effects updates the items in the listbox

Else

MessageBox.Show("Failed to delete user", "Notice")

End If

End If

End Sub

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

Dim classID As Integer = 0

If IsValidEmail(txtEmail.Text) Then

If MessageBox.Show("Are you sure you want to update this record?", "Prompt", MessageBoxButtons.YesNo) = DialogResult.Yes Then

Try

classID = DirectCast(cboClass.SelectedItem, ClassRoom).ClassID

Catch

'This is because the class combo box will only show for students and teachers. So if you're trying to update

'an admins it will give an IndexOutOfBounds since the selected index would be -1

End Try

Dim userRank As Rank = CType(cboRank.SelectedIndex, Rank)

If Admin.UpdateAccount(CInt(txtUserID.Text), txtFirstname.Text, txtLastname.Text, txtEmail.Text, txtPassword.Text, picUser.Image.ToBytes(), classID, userRank) Then

MessageBox.Show("User updated successfully", "Notice")

frmAdmin.btnSearch.PerformClick()

Else

MessageBox.Show("There was an error trying to edit user information", "Notice")

End If

End If

Else

MessageBox.Show("Invalid email address, please enter a valid email address!")

End If

End Sub

Private Function IsValidEmail(email As String) As Boolean

Const emailPattern As String = "[\w\d-+\_.]+@[\w.-]+\.[\w]{2,5}"

Dim isMatch As Boolean = Regex.Match(email, emailPattern).Value = email

Return isMatch

'The email in the parameters is validated using the Regex expression.

'If the Regex engine mathes the email the the function returns true, and if it doesn't

'it returns false

End Function

Private Sub cboRank\_SelectedIndexChanged(sender As Object, e As EventArgs) Handles cboRank.SelectedIndexChanged

'Converts the selectedItem into the Rank Enum.

If CType(cboRank.SelectedIndex, Rank) = Rank.Admin Then

ChangeVisibility(False)

Else

ChangeVisibility(True)

End If

End Sub

Private Sub ChangeVisibility(Value As Boolean)

lblClass.Visible = Value

cboClass.Visible = Value

End Sub

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

Try

Using ofd As New OpenFileDialog With {.Filter = "Image|\*jpg;\*png;\*bmp"}

If ofd.ShowDialog = DialogResult.OK Then picUser.Image = Image.FromFile(ofd.FileName)

End Using

Catch ex As Exception

MessageBox.Show(ex.Message)

End Try

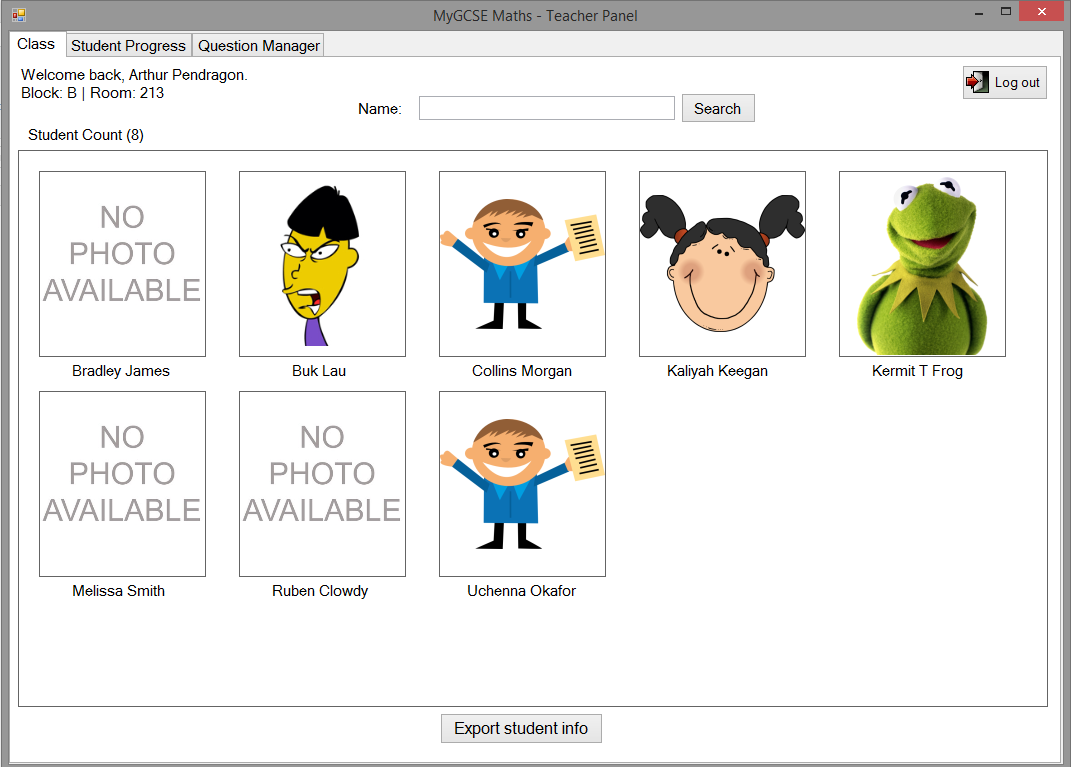
End Sub

End Class

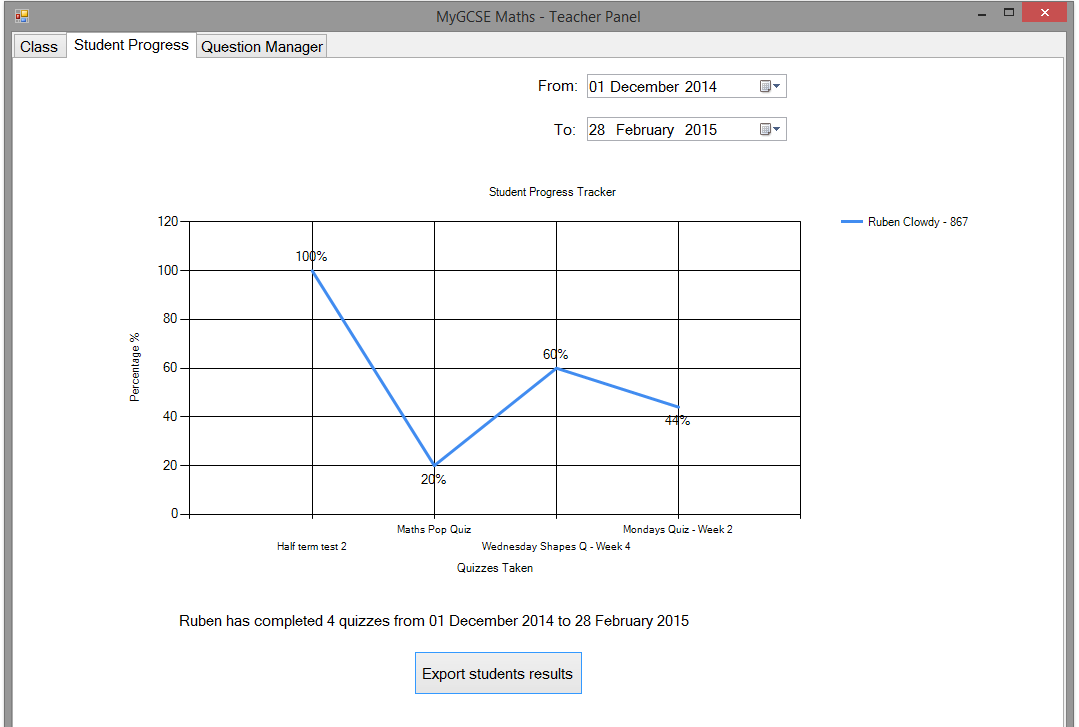
**Title:** frmTeacher.vb

|  |  |
| --- | --- |
| **Description:** | This is the teacher panel. This form is only accessible by teachers. There are 3 tabs to the teacher panel. |

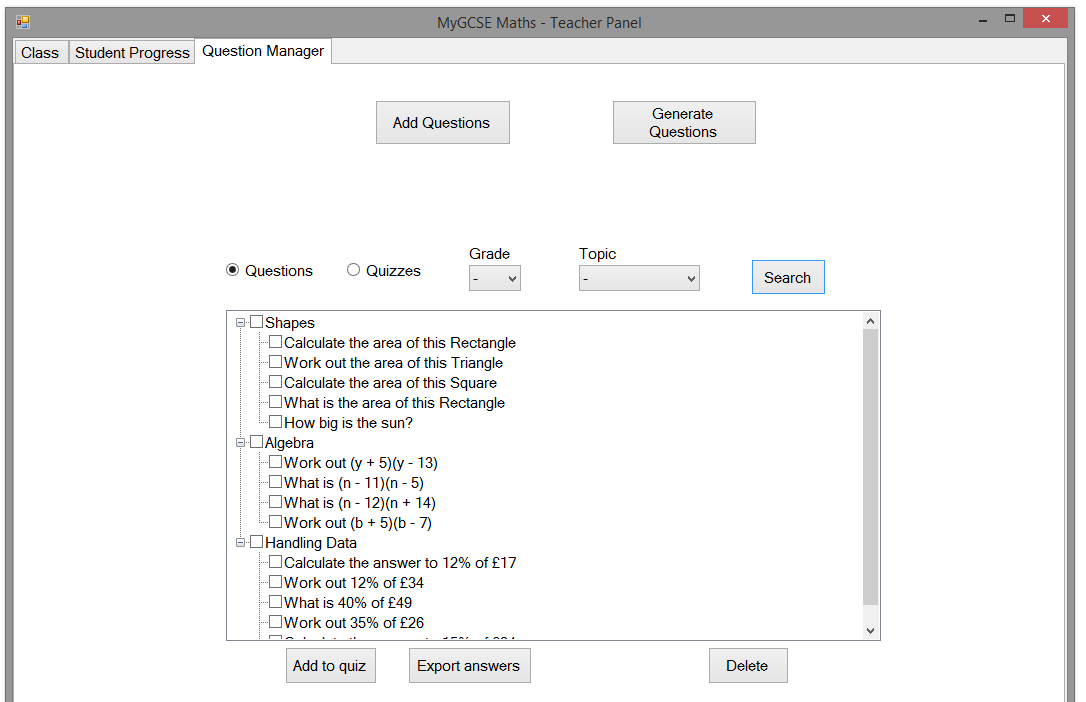
|  |  |
| --- | --- |
| **Tab 1- ClassTab:** | The class tab is where the teacher can view the students in their class. They search, and select which student they want to govern. From there, they can then see the performance for all the quizzes the student has taken. |



|  |  |
| --- | --- |
| **Tab 2 - StudentProgress** | This tab is for viewing the progress of a student. The teacher would select a student from the class tab. Then this tab will load a graph that will show the performance of the student from a specified time period.  The Export student results button is for exporting a .txt file which contains a list of the average mark each student in the teacher’s class has achieved across all completed quizzes. The list is then sorted using the QuickSort algorithm class before saved to a .txt file. |



|  |  |
| --- | --- |
| **Tab 3 – Question Manager** | This tab is for managing questions. This is how the teacher could add or edit questions, or have the system generate questions. Also, the teacher could also add quizzes and manipulate questions that are in the system. |



Imports MyGCSE\_Maths.EntityBase

Imports System.Text

Imports System.Text.RegularExpressions

Imports System.Data.SqlClient

Imports System.Threading

Public Class frmTeacher

Private Teacher As New TeacherService

#Region "Global controls declarations"

Friend WithEvents studentsView As New Panel

Friend WithEvents btnLogout As New Button

Friend WithEvents lblStudentCount As New Label

Friend WithEvents lblWelcomeBack As New Label

Friend WithEvents btnExportInfo As New Button

Friend WithEvents btnSearch As New Button

Friend WithEvents txtSearchField As New TextBox

Private Sub CreateTab1Controls()

btnLogout.Font = New Font("Microsoft Sans Serif", 9.75!, FontStyle.Regular)

btnLogout.Image = My.Resources.logout\_img

btnLogout.ImageAlign = ContentAlignment.MiddleLeft

btnLogout.Location = New Point(949, 7)

btnLogout.Size = New Size(86, 35)

btnLogout.Text = "Log out"

btnLogout.TextAlign = ContentAlignment.MiddleRight

btnLogout.UseVisualStyleBackColor = True

lblWelcomeBack.AutoSize = True

lblWelcomeBack.Location = New Point(5, 7)

lblWelcomeBack.Text = "Welcome back, "

Dim lblName As New Label

lblName.AutoSize = True

lblName.Location = New Point(342, 41)

lblName.Text = "Name:"

txtSearchField.Location = New Point(406, 38)

txtSearchField.Size = New Size(256, 24)

btnSearch.Location = New Point(668, 35)

btnSearch.Size = New Size(75, 30)

btnSearch.Text = "Search"

btnSearch.UseVisualStyleBackColor = True

lblStudentCount.AutoSize = True

lblStudentCount.Location = New Point(12, 67)

lblStudentCount.Text = "Student Count (0)"

studentsView.AutoScroll = True

studentsView.BackColor = Color.Transparent

studentsView.BorderStyle = BorderStyle.FixedSingle

studentsView.Location = New Point(5, 92)

studentsView.Size = New Size(1030, 557)

btnExportInfo.Font = New Font("Microsoft Sans Serif", 12.0!, FontStyle.Regular)

btnExportInfo.Location = New Point(427, 655)

btnExportInfo.Size = New Size(163, 31)

btnExportInfo.Text = "Export student info"

btnExportInfo.UseVisualStyleBackColor = True

Dim ctrlCollection As New Collection From {studentsView, btnLogout, lblStudentCount, lblWelcomeBack, btnExportInfo, lblName, btnSearch, txtSearchField}

For Each ctrl As Control In ctrlCollection

Me.tabMngClass.Controls.Add(ctrl)

Next

End Sub

Friend WithEvents DateTimeTo As New DateTimePicker

Friend WithEvents DateTimeFrom As New DateTimePicker

Friend WithEvents btnExportResult As New Button

Friend WithEvents lblQuizInfo As New Label

Private Sub CreateTab2Controls()

Dim lblTo As New Label

lblTo.AutoSize = True

lblTo.Location = New Point(535, 61)

lblTo.Text = "To:"

DateTimeTo.Location = New Point(571, 58)

DateTimeTo.Size = New Size(200, 24)

Dim lblFrom As New Label

lblFrom.AutoSize = True

lblFrom.Location = New Point(519, 17)

lblFrom.Text = "From:"

DateTimeFrom.Location = New Point(571, 15)

DateTimeFrom.Size = New Size(200, 24)

lblQuizInfo.AutoSize = True

lblQuizInfo.Location = New Point(160, 552)

lblQuizInfo.Text = "....."

btnExportResult.Location = New Point(398, 592)

btnExportResult.Size = New Size(169, 44)

btnExportResult.Text = "Export students results"

Dim ctrlCollection As New Collection From {btnExportResult, lblTo, lblFrom, DateTimeTo, DateTimeFrom, lblQuizInfo}

For Each ctrl As Control In ctrlCollection

Me.tabProgress.Controls.Add(ctrl)

Next

End Sub

Friend WithEvents btnGenerateQuestion As New Button

Friend WithEvents btnAddQuestion As New Button

Friend WithEvents btnSearchQuestion As New Button

Friend WithEvents tvQuestions As New TreeView

Friend WithEvents rdoQuiz As New RadioButton

Friend WithEvents rdoQuestions As New RadioButton

Friend WithEvents cboGrade As New ComboBox

Friend WithEvents cboTopic As New ComboBox

Friend WithEvents btnAddQuiz As New Button

Friend WithEvents btnExportAnswers As New Button

Friend WithEvents btnDeleteQuestion As New Button

Private Sub CreateTab3Controls()

btnAddQuestion.Location = New Point(358, 35)

btnAddQuestion.Size = New Size(136, 45)

btnAddQuestion.Text = "Add Questions"

btnGenerateQuestion.Location = New Point(595, 35)

btnGenerateQuestion.Size = New Size(145, 45)

btnGenerateQuestion.Text = "Generate Questions"

rdoQuestions.AutoSize = True

rdoQuestions.Location = New Point(209, 194)

rdoQuestions.Text = "Questions"

rdoQuiz.AutoSize = True

rdoQuiz.Location = New Point(330, 194)

rdoQuiz.Text = "Quizzes"

Dim lblTopic As New Label

lblTopic.AutoSize = True

lblTopic.Location = New Point(559, 179)

lblTopic.Text = "Topic"

cboTopic.DropDownStyle = ComboBoxStyle.DropDownList

cboTopic.Items.AddRange({"-", "Shapes", "Numbers", "Algebra", "Handling Data"})

cboTopic.Location = New Point(562, 200)

cboTopic.Size = New Size(121, 26)

Dim lblGrade As New Label

lblGrade.AutoSize = True

lblGrade.Location = New Point(449, 179)

lblGrade.Text = "Grade"

cboGrade.DropDownStyle = ComboBoxStyle.DropDownList

cboGrade.Items.AddRange({"-", "A", "B", "C", "D", "E"})

cboGrade.Location = New Point(452, 200)

cboGrade.Size = New Size(52, 26)

btnSearchQuestion.Location = New Point(734, 194)

btnSearchQuestion.Size = New Size(75, 36)

btnSearchQuestion.Text = "Search"

tvQuestions.CheckBoxes = True

tvQuestions.Location = New Point(209, 245)

tvQuestions.Size = New Size(655, 331)

btnAddQuiz.Location = New Point(268, 582)

btnAddQuiz.Size = New Size(92, 37)

btnAddQuiz.Text = "Add to quiz"

btnExportAnswers.Location = New Point(391, 582)

btnExportAnswers.Size = New Size(124, 37)

btnExportAnswers.Text = "Export answers"

btnDeleteQuestion.Location = New Point(691, 582)

btnDeleteQuestion.Size = New Size(81, 37)

btnDeleteQuestion.Text = "Delete"

Dim ctrlCollection As New Collection From {btnAddQuiz, btnSearchQuestion, tvQuestions, cboGrade, cboTopic, lblTopic, lblGrade, btnDeleteQuestion, rdoQuiz, rdoQuestions, btnExportAnswers, btnGenerateQuestion, btnAddQuestion}

For Each ctrl As Control In ctrlCollection

Me.tabQuestions.Controls.Add(ctrl)

Next

End Sub

Sub New()

' This call is required by the designer.

InitializeComponent()

' Add any initialization after the InitializeComponent() call.

CreateTab1Controls()

CreateTab2Controls()

CreateTab3Controls()

End Sub

#End Region

Private Sub frmTeacher\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

lblWelcomeBack.Text = String.Format("Welcome back, {0} {1}.", MyUser.Firstname, MyUser.Lastname)

lblWelcomeBack.Text += String.Format("{0}{1}", Environment.NewLine, Teacher.GetClassInfo(MyUser.ClassID))

InitializeChart()

cboGrade.SelectedIndex = 0

cboTopic.SelectedIndex = 0

rdoQuestions.Checked = True

End Sub

Private Sub Form\_Closing(sender As Object, e As EventArgs) Handles MyBase.FormClosing

Application.Exit()

End Sub

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

Teacher.PromptLogout()

End Sub

Dim FirstTime As Boolean

Dim PreviousState As FormWindowState

Private Sub frmTeacher\_Resize(sender As Object, e As EventArgs) Handles Me.Resize

'Relocates every UI control on this form when resized.

If PreviousState <> FormWindowState.Minimized Then

If FirstTime Then

Select Case Me.WindowState

Case FormWindowState.Normal

ReLocateCtrls(-150)

Case FormWindowState.Maximized

ReLocateCtrls(+150)

End Select

End If

End If

PreviousState = Me.WindowState

FirstTime = True

End Sub

Private Sub ReLocateCtrls(xLocation As Integer)

'When the form is resized, this will resize either shift all the controls to the right of left.

'We can also pass down negative numbers, therefore shifting the position to the left

For Each ctrl As Control In Me.tabMngClass.Controls

ctrl.Location = New Point(ctrl.Location.X + xLocation, ctrl.Location.Y)

Next

For Each ctrl As Control In Me.tabQuestions.Controls

ctrl.Location = New Point(ctrl.Location.X + xLocation, ctrl.Location.Y)

Next

For Each ctrl As Control In Me.tabProgress.Controls

ctrl.Location = New Point(ctrl.Location.X + xLocation, ctrl.Location.Y)

Next

End Sub

Private Sub InitializeChart()

Chart1.Titles.Add("Student Progress Tracker")

Chart1.ChartAreas(0).AxisY.Title = "Percentage %"

Chart1.ChartAreas(0).AxisX.Title = "Quizzes Taken"

Chart1.Series(0).IsValueShownAsLabel = True

Chart1.Series(0).LabelFormat = "{#'%'}"

End Sub

Private Sub SaveToTxt(content As String)

'Saves the content of the string password by the paramater

Using sfd As New SaveFileDialog With {.Filter = "Text File(.txt) |\*.txt"}

If sfd.ShowDialog = DialogResult.OK Then

IO.File.WriteAllText(sfd.FileName, content)

Process.Start(sfd.FileName)

End If

End Using

End Sub

#Region "Class Tab"

Private Sub txtSearchField\_KeyDown(sender As Object, e As KeyEventArgs) Handles txtSearchField.KeyDown

If e.KeyCode = Keys.Enter Then btnSearch.PerformClick()

End Sub

Private Sub ClearControls()

Do

For Each ctrl As Control In studentsView.Controls

ctrl.Dispose()

studentsView.Controls.Remove(ctrl)

Next

Loop Until studentsView.Controls.Count = 0

End Sub

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

ClearControls()

Dim studentList As List(Of User) = Teacher.GetStudents(MyUser.ClassID, txtSearchField.Text)

If IsNothing(studentList) Then

MessageBox.Show("User not found")

lblStudentCount.Text = String.Format("Student Count ({0})", 0)

Else

'The thread pool multi threads this. This is so it's fast. And since the algorithm can be a

'bit CPU intensive.

ThreadPool.QueueUserWorkItem(Sub() CreateDynamicStudents(studentList))

lblStudentCount.Text = String.Format("Student Count ({0}) ", studentList.Count)

End If

End Sub

Private Sub CreateDynamicStudents(students As List(Of User))

ClearControls() 'Clears all the pre-exiting controls in the panel first.

Invoke(Sub() btnSearch.Enabled = False)

'Initial locations

Dim xLocation As Integer = 20

Dim yLocation As Integer = 20

Dim amountCreated As Integer = 0

For i = 0 To students.Count - 1

Dim picBox As New CustomPictureBox

'Instanciate the CustomPictureBox object.

'Defines all its properties.

picBox.Name = students(i).Firstname & " " & students(i).Lastname

picBox.UserID = students(i).UserID

picBox.Username = students(i).Username

picBox.Password = students(i).Password

picBox.Size = New Size(167, 186)

picBox.Location = New Point(xLocation, yLocation)

picBox.BorderStyle = BorderStyle.FixedSingle

picBox.SizeMode = PictureBoxSizeMode.Zoom

picBox.Cursor = Cursors.Hand

picBox.Image = Image.FromStream(students(i).Image.ToStream())

CreateDynamicLabels(picBox.Name, xLocation, yLocation)

'Creates the label of the student the picturebox represents underneath the picturebox

amountCreated += 1

xLocation += 200

'Checks the amount created. If it's 5. The it changes it increments the yLocation.

'Which moves it down, and resets the xLocation to the left of the Parent Container.

If amountCreated = 5 Then

xLocation = 20

yLocation += 220

amountCreated = 0

End If

AddHandler picBox.Click, AddressOf StudentPicture\_Clicked

Invoke(Sub() Me.studentsView.Controls.Add(picBox))

'Adds the object to the Panel1.Controls.

'We also add a click event handler.

'We're invoking the methods because we're running this on a seperate thread'

'And to make a safetly call we need to Invoke the call using a delegate sub.

'Or we'd get a CrossThreadCallExcpetion thrown.

Next

Invoke(Sub() Me.studentsView.Focus())

Invoke(Sub() btnSearch.Enabled = True)

End Sub

Private Sub CreateDynamicLabels(name As String, xLocation As Integer, yLocation As Integer)

'Creates the label of the student

Dim lblStudentName As New Label With {.Text = name, .AutoSize = True}

xLocation += GetEstimateX(lblStudentName)

yLocation += 190

lblStudentName.Location = New Point(xLocation, yLocation)

Invoke(Sub() Me.studentsView.Controls.Add(lblStudentName))

End Sub

Private Function GetEstimateX(lbl As Label) As Integer

'This provides an estimate of the xLocation of the label.

'In order to place the label in the right place.

Dim amount As Integer

'Measures the labels width

Select Case MeasureLabel(lbl).Width

Case Is <= 30 : amount = 60

Case Is <= 40 : amount = 55

Case Is <= 50 : amount = 50

Case Is <= 60 : amount = 45

Case Is <= 65 : amount = 40

Case Is <= 70 : amount = 35

Case Is <= 80 : amount = 30

Case Is <= 100 : amount = 25

Case Is >= 101 : amount = 5

End Select

'amount is just a guess of how much to the right the xLocation should be moved to

Return amount

End Function

Private Function MeasureLabel(lbl As Label) As SizeF

'Measures the label and returns the size

Return Me.CreateGraphics.MeasureString(lbl.Text, lbl.Font)

End Function

Private Function GetMidPoint(p As PictureBox) As Point

'The size of the picturebox == p.Size = New Size(167, 186) (Width, Height)

'So to get the co-ordinate of the bottom left, we keep the xlocation the same

'to find the yLocation, we move down by the size of the Height of the picutrebox

'That gives us the x & y co-ordinates for the bottom left of the picturebox

'We use the same concept to find the x & y co-ordinates for the bottom right co-ordinate

'Gets the midpoint location of the bottom end of a picturebox.

Dim x1 As Integer = (p.Location.X)

Dim y1 As Integer = (p.Location.Y + p.Size.Height)

Dim x2 As Integer = (p.Location.X + p.Size.Width)

Dim y2 As Integer = (p.Location.Y + p.Size.Height)

Dim bottomLeft As New Point(x1, x2)

Dim bottomRight As New Point(y1, y2)

'Calculates the midpoint then returns it

Dim midPoint As New Point(CInt((x2 + x1) / 2), CInt((y2 + y1) / 2))

Return midPoint

End Function

Private Sub StudentPicture\_Clicked(sender As Object, e As EventArgs)

Dim picbx As CustomPictureBox = DirectCast(sender, CustomPictureBox)

'We convert/cast the sender as the Object CustomPictureBox, this way we can

'Access it's methods and attributes.

Chart1.Series(0).Name = String.Format("{0} - {1}", picbx.Name, picbx.UserID)

DateTimeFrom.Value = FirstDayOfMonth()

DateTime\_ValueChanged(sender, e) 'This is to fix a bug where the DateValueChanged event wouldn't fire

'because the previous DateTimeValue was the same as the new one. So therefore the event wouldn't fire.

'Changing the value of the datetime picker will raise the ValueChanged event

Me.TabControl1.SelectedTab = Me.tabProgress

End Sub

Private Function FirstDayOfMonth() As DateTime

Return New DateTime(Today.Year, Today.Month, 1) 'Gets the first day of the month as date

End Function

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

If studentsView.Controls.OfType(Of CustomPictureBox).Count = 0 Then

MessageBox.Show("There are no students in the panel. Please refine your search")

Else

SaveToTxt(GetStudentsInfo)

End If

End Sub

Private Function GetStudentsInfo() As String

'Loops through each picturebox in the Panel1.Controls.

'Since each picturebox is a CustomPictureBox object, it can access the username,

'password and name of each object directly.

Dim text As New StringBuilder

For Each picbx As CustomPictureBox In studentsView.Controls.OfType(Of PictureBox)()

text.AppendLine("Name: " & picbx.Name)

text.AppendLine("Username: " & picbx.Username)

text.AppendLine("Password: " & picbx.Password)

text.AppendLine()

Next

Return text.ToString

End Function

#End Region

#Region "Students Progress Tab"

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

Dim studentQuizList As List(Of StudentQuizInfo) = Teacher.GetStudentsQuizResults(MyUser.ClassID)

'Gets a sorted list of the overall performance of each student that has completed atleast 1 quiz in the

'teachers class.

Dim sb As New StringBuilder

sb.AppendLine(" MyGCSE Maths Benchmark ")

sb.AppendLine()

For Each student In studentQuizList

sb.AppendLine(String.Format("Name: {0} {1}", student.Firstname, student.Lastname))

sb.AppendLine(String.Format("Average: {0}%", student.Average))

sb.AppendLine(String.Format("Total Quizzes Completed: {0}", student.QuizScores.Count))

sb.AppendLine()

Next

'Saves it to a txt file.

SaveToTxt(sb.ToString())

End Sub

Public Sub DateTime\_ValueChanged(sender As Object, e As EventArgs) Handles DateTimeFrom.ValueChanged, DateTimeTo.ValueChanged

Dim highestDate As Date = Date.Today.AddDays(1)

If DateTimeFrom.Value < #1/1/2014# Then

MessageBox.Show("Date cannot be lower than 1/1/2014")

ElseIf DateTimeTo.Value > highestDate Then

MessageBox.Show("Date cannot be higher than " & highestDate.AddDays(-1))

Else

Dim userId As Integer

'Using regex we parse the userId from the Chart1.Series name

'Which looks like. Thomas Clowdy - 4412

Dim tempStr As String = Regex.Replace(Chart1.Series(0).Name, "[^0-9]", String.Empty)

If Integer.TryParse(tempStr, userId) Then

'Checks if the parse was succesful, because sometimes the Chart1.Series.Name

'May be empty because they haven't selected a student yet.

Dim name As String = Chart1.Series(0).Name.Split(" "c)(0)

Me.Chart1.Series(0).Points.Clear()

Dim studentResults As List(Of Quiz) = Teacher.GetStudentQuizResults(userId, DateTimeFrom.Value, DateTimeTo.Value)

'Gets a List of the QuizObject which contains the students result. and the quiz name etc.

If IsNothing(studentResults) Then

lblQuizInfo.Text = String.Format("No quizzes has been completed by {0} from {1} to {2}", name, DateTimeFrom.Text, DateTimeTo.Text)

Else

'Plots each items percentage and title on the graph

For Each quiz As Quiz In studentResults

Me.Chart1.Series(0).Points.Add(quiz.Percentage).AxisLabel = quiz.Title

Next

lblQuizInfo.Text = String.Format("{0} has completed {1} quizzes from {2} to {3}", name, Chart1.Series(0).Points.Count, DateTimeFrom.Text, DateTimeTo.Text)

End If

End If

End If

End Sub

#End Region

#Region "Question Manager Tab"

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

frmQuestions.UsageMode(ActionType.Add)

'Before we call frmQuestion.Show, we must let it know how we intend to use the form.

'Weather its to Add a question, or edit a question.

frmQuestions.Show()

End Sub

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

tvQuestions.Nodes.Clear()

Dim grade As String = cboGrade.SelectedItem.ToString

Dim topic As String = cboTopic.SelectedItem.ToString

If rdoQuestions.Checked Then

btnAddQuiz.Visible = True

HandleQuestion(topic, grade)

ElseIf rdoQuiz.Checked Then

btnAddQuiz.Visible = False

Teacher.FetchQuiz()

'Fetches all the quizzes and adds them to Questions Tree View

End If

tvQuestions.ExpandAll()

If tvQuestions.Nodes.Count > 0 Then tvQuestions.Nodes(0).EnsureVisible() 'Scrolls up to the top after expanding all

End Sub

Private Sub HandleQuestion(topic As String, grade As String)

'This is used to search for a question by using a combo box.

'It can search based upon a topic and or grade.

'So if the topic 'Algebra' was selected, all the algebra questions will be returned.

'If the topic 'Algebra' and the Grade 'B' was selected then all algebra questions of grade B will be returned.

'If 'C' was selected and nothing for topic was selected then all the C grade questions will be returned.

'Regardless of the topic.

If topic = "-" And grade = "-" Then

For Each itm As String In cboTopic.Items

If itm <> "-" Then Teacher.PopulateTreeView(String.Empty, itm)

Next

ElseIf topic = "-" Or grade = "-" Then

If grade = "-" Then

grade = String.Empty

Teacher.PopulateTreeView(grade, topic)

End If

If topic = "-" Then

For Each itm As String In cboTopic.Items

Teacher.PopulateTreeView(grade, itm)

Next

End If

Else

Teacher.PopulateTreeView(grade, topic)

End If

End Sub

Private Sub tvQuestions\_AfterCheck(sender As Object, e As TreeViewEventArgs) Handles tvQuestions.AfterCheck

'Event is raised after a parent treenode is Checked

For Each childNode As TreeNode In e.Node.Nodes

childNode.Checked = e.Node.Checked

Next

End Sub

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

frmGenerateQuestions.Show()

frmGenerateQuestions.BringToFront()

End Sub

Private Function GetCheckedCount() As Integer

'Checks how many tree nodes were selected and returns the count.

'This is used to validate user actions.

Dim counter As Integer

For Each parentNode As TreeNode In tvQuestions.Nodes

For Each childNode As CustomTreeNode In parentNode.Nodes

If childNode.Checked Then counter += 1

Next

Next

Return counter

End Function

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

'This loops through every checked childNode of every parentNode, and adds them to a quiz.

'Each checked node is a question.

If GetCheckedCount() = 0 Then

MessageBox.Show("You must select atleast one question before continuing")

Else

Dim quizName As String = InputBox(String.Format("Enter the title of the quiz please{0}e.g.Tuesday's Quiz - Week 4", Environment.NewLine, "Quiz Title"))

'Checks if the InputBox content was not empty.

If String.IsNullOrWhiteSpace(quizName) = False Then

'Loops through each childnode of each parentnode.

'And adds

Dim questionIdList As New List(Of Integer)

For Each parentNode As TreeNode In tvQuestions.Nodes

For Each childNode As CustomTreeNode In parentNode.Nodes.OfType(Of CustomTreeNode)()

If childNode.Checked Then questionIdList.Add(childNode.QuestionID)

Next

Next

If Teacher.AddQuiz(questionIdList, quizName) Then MessageBox.Show("Successfully added questions into the quiz set")

End If

End If

End Sub

Private Sub DeleteQstn()

Dim counter As Integer

'Deletes all the selected questions in the TreeView.

For Each parentNode As TreeNode In tvQuestions.Nodes

For Each childNode As CustomTreeNode In parentNode.Nodes

If childNode.Checked Then

If QuestionHelper.DeleteQuestion(childNode.QuestionID) Then counter += 1 'Increments counter if delete successsfull

End If

Next

Next

If counter >= 1 Then

MessageBox.Show("Sucessfully deleted " & counter & " question(s)")

btnSearchQuestion.PerformClick()

End If

End Sub

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

If GetCheckedCount() = 0 Then

MessageBox.Show("You must select atleast one question before continuing")

Else

If MessageBox.Show("Are you sure you want to delete the selected questions?", "Prompt", MessageBoxButtons.YesNo) \_

= DialogResult.Yes Then DeleteQstn()

End If

End Sub

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

If GetCheckedCount() = 0 Then

MessageBox.Show("You must select atleast one question before continuing")

Else

'Uses a class which will return the Question and

'Answer of each question that is passed down by the parameters.

Dim questionIdList As New List(Of Integer)

For Each parentNode As TreeNode In tvQuestions.Nodes

For Each childNode As CustomTreeNode In parentNode.Nodes

If childNode.Checked Then questionIdList.Add(childNode.QuestionID)

Next

Next

'The function returns a format. And we expor that.

SaveToTxt(Teacher.ExportQuestions(questionIdList))

End If

End Sub

Private Sub tvQuestions\_AfterSelect(sender As Object, e As TreeViewEventArgs) Handles tvQuestions.AfterSelect

'When a question from the QuestionsTreeview is selected.

'We cast the object selected to a CustomTreeNode

If e.Node.GetType Is GetType(CustomTreeNode) Then

Dim qID = DirectCast(e.Node, CustomTreeNode).QuestionID

'We stored the casted value at the variable qID

frmQuestions.CurrentQuestionID = qID

Dim question As Question = QuestionHelper.GetQuestion(qID)

'We use the QuestionHelper class to request for a question.

'By asking it to return all it knows about the question with the

'specific questionID

frmQuestions.picQuestion.Image = Image.FromStream(question.Image.ToStream())

frmQuestions.txtQuestion.Text = question.QuestionString

frmQuestions.cboTopic.SelectedItem = question.Topic

frmQuestions.cboGrade.SelectedItem = question.Grade.ToString()

frmQuestions.txtCorrectAnswer.Text = question.Answer

'Set the questions returned to the controls of frmQuestions.

If question.IsMultiChoice Then

frmQuestions.rdoMultiChoice.Checked = True

frmQuestions.txtFalseAnswer1.Text = question.FalseAnswers(0)

frmQuestions.txtFalseAnswer2.Text = question.FalseAnswers(1)

frmQuestions.txtFalseAnswer3.Text = question.FalseAnswers(2)

Else

frmQuestions.rdoSingleAnswer.Checked = True

End If

frmQuestions.UsageMode(ActionType.Edit)

frmQuestions.Show()

End If

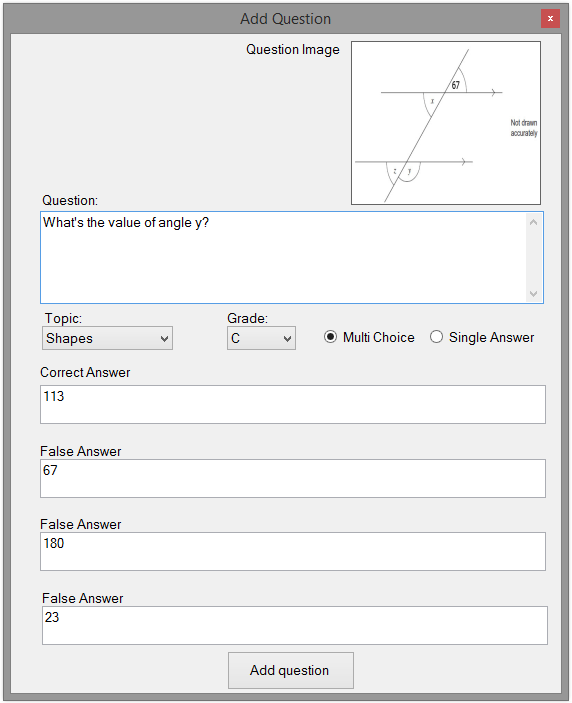
End Sub

#End Region

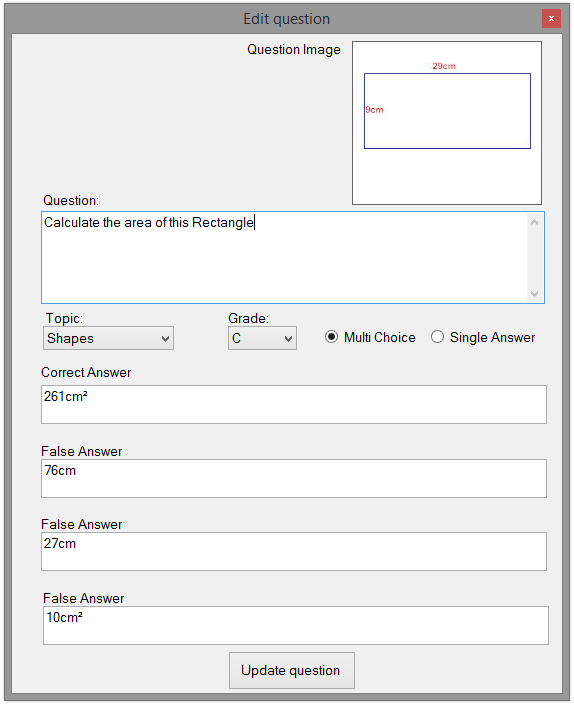
End Class

**Title:** frmQuestions.vb

|  |  |
| --- | --- |
| **Description:** | This form is for manipulating questions. This form can be used both for adding and edtting questions, it can adapt to do different things depending on how the form is called. If the teacher clicks the add button it will be used for adding questions, if the teacher selects a pre-existing question from the questions tree view the form will change to edit mode therefore allowing you to edit the question. |



This screenshot shows how the same form can be used for both adding and editing a question. It all depends on how the form is called.



Imports MyGCSE\_Maths.EntityBase

Public Class frmQuestions

Public CurrentAction As ActionType

Public CurrentQuestionID As Integer

#Region "Global controls declerations"

Friend WithEvents lblFalseAnswer3 As New Label

Friend WithEvents lblFalseAnswer2 As New Label

Friend WithEvents lblFalseAnswer1 As New Label

Friend WithEvents txtFalseAnswer1 As New TextBox

Friend WithEvents txtFalseAnswer2 As New TextBox

Friend WithEvents txtFalseAnswer3 As New TextBox

Friend WithEvents txtCorrectAnswer As New TextBox

Friend WithEvents rdoSingleAnswer As New RadioButton

Friend WithEvents rdoMultiChoice As New RadioButton

Friend WithEvents picQuestion As New PictureBox

Friend WithEvents txtQuestion As New TextBox

Friend WithEvents btn As New Button

Friend WithEvents cboGrade As New ComboBox

Friend WithEvents cboTopic As New ComboBox

Private Sub CreateControls()

Dim lblQuestion As New Label

lblQuestion.AutoSize = True

lblQuestion.Font = New Font("Microsoft Sans Serif", 9.75!, FontStyle.Regular)

lblQuestion.Location = New Point(28, 158)

lblQuestion.Text = "Question:"

Dim lblQstnImage As New Label

lblQstnImage.AutoSize = True

lblQstnImage.Font = New Font("Microsoft Sans Serif", 9.75!, FontStyle.Regular)

lblQstnImage.Location = New Point(232, 7)

lblQstnImage.Text = "Question Image"

picQuestion.BorderStyle = BorderStyle.FixedSingle

picQuestion.Cursor = Cursors.Hand

picQuestion.Location = New Point(340, 7)

picQuestion.Size = New Size(190, 164)

picQuestion.SizeMode = PictureBoxSizeMode.StretchImage

txtQuestion.Font = New Font("Microsoft Sans Serif", 9.75!, FontStyle.Regular)

txtQuestion.Location = New Point(29, 177)

txtQuestion.MaxLength = 100

txtQuestion.Multiline = True

txtQuestion.ScrollBars = ScrollBars.Vertical

txtQuestion.Size = New Size(504, 93)

Dim lblTopic As New Label

lblTopic.AutoSize = True

lblTopic.Font = New Font("Microsoft Sans Serif", 9.75!, FontStyle.Regular)

lblTopic.Location = New Point(30, 276)

lblTopic.Text = "Topic:"

cboTopic.DropDownStyle = ComboBoxStyle.DropDownList

cboTopic.Font = New Font("Microsoft Sans Serif", 9.75!, FontStyle.Regular)

cboTopic.Items.AddRange({"Numbers", "Algebra", "Shapes", "Handling Data"})

cboTopic.Location = New Point(31, 292)

cboTopic.Size = New Size(131, 24)

Dim lblGrade As New Label

lblGrade.AutoSize = True

lblGrade.Font = New Font("Microsoft Sans Serif", 9.75!, FontStyle.Regular)

lblGrade.Location = New Point(213, 276)

lblGrade.Text = "Grade:"

cboGrade.DropDownStyle = ComboBoxStyle.DropDownList

cboGrade.Font = New Font("Microsoft Sans Serif", 9.75!, FontStyle.Regular)

cboGrade.Items.AddRange({"A", "B", "C", "D", "E"})

cboGrade.Location = New Point(216, 292)

cboGrade.Size = New Size(69, 24)

rdoMultiChoice.AutoSize = True

rdoMultiChoice.Font = New Font("Microsoft Sans Serif", 9.75!, FontStyle.Regular)

rdoMultiChoice.Location = New Point(313, 293)

rdoMultiChoice.Text = "Multi Choice"

rdoSingleAnswer.AutoSize = True

rdoSingleAnswer.Font = New Font("Microsoft Sans Serif", 9.75!, FontStyle.Regular)

rdoSingleAnswer.Location = New Point(419, 293)

rdoSingleAnswer.Text = "Single Answer"

Dim lblCorrectAnswer As New Label

lblCorrectAnswer.AutoSize = True

lblCorrectAnswer.Font = New Font("Microsoft Sans Serif", 9.75!, FontStyle.Regular)

lblCorrectAnswer.Location = New Point(26, 330)

lblCorrectAnswer.Text = "Correct Answer"

txtCorrectAnswer.Font = New Font("Microsoft Sans Serif", 9.75!, FontStyle.Regular)

txtCorrectAnswer.Location = New Point(29, 351)

txtCorrectAnswer.Multiline = True

txtCorrectAnswer.Size = New Size(506, 39)

lblFalseAnswer1.AutoSize = True

lblFalseAnswer1.Font = New Font("Microsoft Sans Serif", 9.75!, FontStyle.Regular)

lblFalseAnswer1.Location = New Point(26, 409)

lblFalseAnswer1.Text = "False Answer"

txtFalseAnswer1.Font = New Font("Microsoft Sans Serif", 9.75!, FontStyle.Regular)

txtFalseAnswer1.Location = New Point(29, 425)

txtFalseAnswer1.Multiline = True

txtFalseAnswer1.Size = New Size(506, 39)

lblFalseAnswer2.AutoSize = True

lblFalseAnswer2.Font = New Font("Microsoft Sans Serif", 9.75!, FontStyle.Regular)

lblFalseAnswer2.Location = New Point(26, 482)

lblFalseAnswer2.Text = "False Answer"

txtFalseAnswer2.Font = New Font("Microsoft Sans Serif", 9.75!, FontStyle.Regular)

txtFalseAnswer2.Location = New Point(29, 498)

txtFalseAnswer2.Multiline = True

txtFalseAnswer2.Size = New Size(506, 39)

lblFalseAnswer3.AutoSize = True

lblFalseAnswer3.Font = New Font("Microsoft Sans Serif", 9.75!, FontStyle.Regular)

lblFalseAnswer3.Location = New Point(28, 556)

lblFalseAnswer3.Text = "False Answer"

txtFalseAnswer3.Font = New Font("Microsoft Sans Serif", 9.75!, FontStyle.Regular)

txtFalseAnswer3.Location = New Point(31, 572)

txtFalseAnswer3.Multiline = True

txtFalseAnswer3.Size = New Size(506, 39)

btn.Font = New Font("Microsoft Sans Serif", 9.75!, FontStyle.Regular)

btn.Location = New Point(216, 617)

btn.Size = New Size(128, 39)

btn.UseVisualStyleBackColor = True

Dim ctrlCollection As New Collection From {lblFalseAnswer3, lblFalseAnswer2, lblFalseAnswer1, lblCorrectAnswer, txtFalseAnswer2, txtFalseAnswer3, txtFalseAnswer1, txtCorrectAnswer, cboGrade, cboTopic, lblQstnImage, lblTopic, lblGrade, rdoSingleAnswer, rdoMultiChoice, picQuestion, txtQuestion, lblQuestion, btn}

For Each ctrl As Control In ctrlCollection

Me.Controls.Add(ctrl)

Next

End Sub

Sub New()

' This call is required by the designer.

InitializeComponent()

' Add any initialization after the InitializeComponent() call.

CreateControls()

End Sub

#End Region

Private Sub frmQuestions\_Shown(sender As Object, e As EventArgs) Handles MyBase.Shown

Me.BringToFront()

End Sub

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

ChangeVisibility(False)

End Sub

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

ChangeVisibility(True)

End Sub

Public Sub ChangeVisibility(Value As Boolean)

'Changes the visibily of certain text boxes depeind on which radio button is checked.

txtFalseAnswer1.Visible = Value

txtFalseAnswer2.Visible = Value

txtFalseAnswer3.Visible = Value

lblFalseAnswer1.Visible = Value

lblFalseAnswer2.Visible = Value

lblFalseAnswer3.Visible = Value

End Sub

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

Try

Using ofd As New OpenFileDialog With {.Filter = "Image File |\*.jpg; \*.gif; \*.png"}

If ofd.ShowDialog = DialogResult.OK Then picQuestion.Image = Image.FromFile(ofd.FileName)

End Using

Catch ex As Exception

MessageBox.Show(ex.Message)

End Try

End Sub

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

If InputEmpty() = False Then

If CurrentAction = ActionType.Add Then

If MessageBox.Show("Are you sure you want to add this question?", "Prompt", MessageBoxButtons.YesNo) = DialogResult.Yes Then

AddQuestion()

End If

ElseIf CurrentAction = ActionType.Edit Then

If MessageBox.Show("Are you sure you want to update this question?", "Prompt", MessageBoxButtons.YesNo) = DialogResult.Yes Then

EditQuestion()

End If

End If

Else

MessageBox.Show("Cannot leave any of the input boxes empty")

End If

End Sub

Private Function InputEmpty() As Boolean

'This Function checks too see if any of the controls have been left empty.

'If Single Answer Radio button is checked, this checks if the appropriate controls

'Are set or empty, and if it, it returns true. Same with Multi Answer radio button.

'If it checks everything and its all NOT empty, it returns false meaning the controls are not

'Empty.

If rdoSingleAnswer.Checked Then

If String.IsNullOrWhiteSpace(txtQuestion.Text) Or String.IsNullOrWhiteSpace(txtCorrectAnswer.Text) \_

Or cboGrade.SelectedIndex = -1 Or cboTopic.SelectedIndex = -1 Then

Return True

End If

ElseIf rdoMultiChoice.Checked Then

For Each txtbx As TextBox In Me.Controls.OfType(Of TextBox)()

If String.IsNullOrWhiteSpace(txtbx.Text) Then Return True

Next

If cboGrade.SelectedIndex = -1 Or cboTopic.SelectedIndex = -1 Then Return True

Else

Return True

End If

Return False

End Function

Private Sub FormatText() Handles txtCorrectAnswer.TextChanged, txtFalseAnswer1.TextChanged,

txtFalseAnswer2.TextChanged, txtFalseAnswer3.TextChanged, txtQuestion.TextChanged

' Loops through each textbox on the form, and loops through each key/item in the dictionary

' and if the current textbox contains the text in the dictionary, it replaces it with value of the key in the dictionary

Dim unicode As New Dictionary(Of String, String) From {{"^2", "² "}, {"^3", "³ "}, {"1/2", "½ "}, {"3/4", "¾ "}, {"1/4", "¼ "}}

For Each TextBx As TextBox In Me.Controls.OfType(Of TextBox)()

For Each x As String In unicode.Keys

If TextBx.Text.Contains(x) Then

TextBx.Text = TextBx.Text.Replace(x, unicode(x))

TextBx.Select(TextBx.Text.Length, 0)

End If

Next

Next

End Sub

Private Function GetCurrentQuestion() As Question

Dim q As New Question

q.FalseAnswers = {txtFalseAnswer1.Text.Trim, txtFalseAnswer2.Text.Trim, txtFalseAnswer3.Text.Trim}

' Put the false answers into an array

q.Topic = cboTopic.SelectedItem.ToString()

q.Grade = CChar(cboGrade.SelectedItem)

q.QuestionString = txtQuestion.Text.Trim()

q.Image = picQuestion.Image.ToBytes()

q.Answer = txtCorrectAnswer.Text

q.IsMultiChoice = rdoMultiChoice.Checked

Return q

End Function

Private Sub AddQuestion()

Dim q As Question = GetCurrentQuestion()

If QuestionHelper.AddQuestion(q) = True Then

MessageBox.Show("Question has been added succesfully!")

Else

MessageBox.Show("An error occured whilst to adding a new question")

End If

End Sub

Private Sub EditQuestion()

Dim q As Question = GetCurrentQuestion()

If QuestionHelper.UpdateQuestion(CurrentQuestionID, q) Then

MessageBox.Show("Question updated sucessfully!")

Else

MessageBox.Show("Error updating question")

End If

End Sub

Public Sub UsageMode(action As ActionType)

'The form adapts itself depending on they action type.

If action = ActionType.Add Then

CurrentAction = action

btn.Text = "Add question"

Me.Text = "Add Question"

ElseIf action = ActionType.Edit Then

CurrentAction = action

btn.Text = "Update question"

Me.Text = "Edit question"

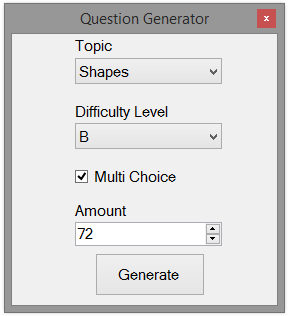
End If

End Sub

End Class

**Title:** frmGenerateQuestions.vb

|  |  |
| --- | --- |
| **Description:** | This form is the UI for generating questions. The criteria is filled in and when the generate button is clicked, it will generate questions based on the criteria. This form can only be accessed by the teacher. |



Imports MyGCSE\_Maths.EntityBase

Public Class frmGenerateQuestions

#Region "Global controls declerations"

Friend WithEvents cboTopic As New ComboBox

Friend WithEvents cboGrade As New ComboBox

Friend WithEvents btnGenerate As New Button

Friend WithEvents nudAmount As New NumericUpDown

Friend WithEvents chkboxMultiChoice As New CheckBox

Private Sub CreateControls()

Dim lblTopic As New Label

lblTopic.AutoSize = True

lblTopic.Font = New Font("Microsoft Sans Serif", 11.25!, FontStyle.Regular)

lblTopic.Location = New Point(60, 2)

lblTopic.Text = "Topic"

cboTopic.DropDownStyle = ComboBoxStyle.DropDownList

cboTopic.Items.AddRange({"Numbers", "Shapes", "Algebra", "Handling Data"})

cboTopic.Location = New Point(63, 24)

cboTopic.Size = New Size(147, 26)

Dim lblGrade As New Label

lblGrade.AutoSize = True

lblGrade.Font = New Font("Microsoft Sans Serif", 11.25!, FontStyle.Regular)

lblGrade.Location = New Point(60, 68)

lblGrade.Text = "Difficulty Level"

cboGrade.DropDownStyle = ComboBoxStyle.DropDownList

cboGrade.Items.AddRange({"A", "B", "C", "D", "E"})

cboGrade.Location = New Point(63, 89)

cboGrade.Size = New Size(147, 26)

btnGenerate.Font = New Font("Microsoft Sans Serif", 11.25!, FontStyle.Regular)

btnGenerate.Location = New Point(83, 219)

btnGenerate.Size = New Size(110, 43)

btnGenerate.Text = "Generate"

chkboxMultiChoice.AutoSize = True

chkboxMultiChoice.Location = New Point(63, 132)

chkboxMultiChoice.Text = "Multi Choice"

Dim lblAmount As New Label

lblAmount.AutoSize = True

lblAmount.Font = New Font("Microsoft Sans Serif", 11.25!, FontStyle.Regular)

lblAmount.Location = New Point(60, 167)

lblAmount.Text = "Amount"

nudAmount.Location = New Point(63, 188)

nudAmount.Size = New Size(147, 24)

Dim ctrlCollection As New Collection From {cboTopic, lblTopic, lblGrade, btnGenerate, cboGrade, nudAmount, lblAmount, chkboxMultiChoice}

For Each ctrl As Control In ctrlCollection

Me.Controls.Add(ctrl)

Next

End Sub

Sub New()

' This call is required by the designer.

InitializeComponent()

' Add any initialization after the InitializeComponent() call.

CreateControls()

End Sub

#End Region

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

If nudAmount.Value > 0 And cboTopic.SelectedIndex > -1 And cboGrade.SelectedIndex > -1 Then

'Validates to check if the criteron is all filled in.

Dim promptMsg As String = String.Format("Are you sure you want to generate {0} {1} question(s)", nudAmount.Value, cboTopic.SelectedItem)

'Prompts the user if they want to generate the questions.

If MessageBox.Show(promptMsg, "Notice", MessageBoxButtons.YesNo) = DialogResult.Yes Then

Dim topic As Topic = CType(cboTopic.SelectedIndex, Topic)

Dim grade As Char = CChar(cboGrade.SelectedItem)

Dim amountRequested As Integer = CInt(nudAmount.Value)

Dim questionsList As List(Of Question) = QuestionGenerator.GenerateQuestions(topic, grade, amountRequested, chkboxMultiChoice.Checked)

'Calls the GenerateQuestion() from the question generator class.

'Passes down the topic it wants to generate, grade, and amount in the parameters.

'The function then returns a list/array of the questions generated.

Dim counter As Integer = 0

For Each question As Question In questionsList

If QuestionHelper.AddQuestion(question) Then counter += 1

Next

'A for loop to add each question in the questionsList to the database using the

'QuestionHelper class.

Dim str As String = String.Format("{0} {1} questions have been generated and added sucessfully!", counter, topic.ToString())

MessageBox.Show(str)

'success message shown

End If

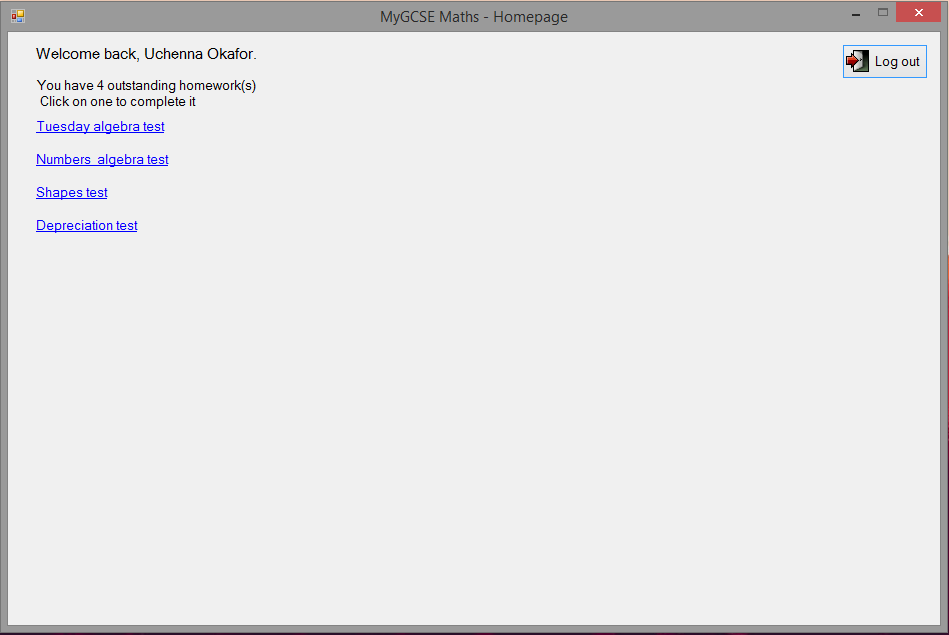
Else

MessageBox.Show("You must not leave any of the options empty")

End If

End Sub

End Class

**Title:** frmStudent.vb

|  |  |
| --- | --- |
| **Description:** | This is the student form. This is where the student goes to complete outstanding quizzes set by their teacher. The student also gets immediate result of their performance, which is also sent to their teacher. |

Imports MyGCSE\_Maths.EntityBase

Public Class frmStudent

Private Student As New StudentService

#Region "Global controls decelrations"

Friend WithEvents lblWelcomeBack As New Label

Friend WithEvents btnLogout As New Button

Friend WithEvents lblHomework As New Label

Friend WithEvents gpQuizQuestion As New GroupBox

Friend WithEvents lblQuestion As New Label

Friend WithEvents picQuestion As New PictureBox

Friend WithEvents rdoFalseAnswer1 As New RadioButton

Friend WithEvents rdoFalseAnswer2 As New RadioButton

Friend WithEvents rdoFalseAnswer3 As New RadioButton

Friend WithEvents rdoCorrectAnswer As New RadioButton

Friend WithEvents txtAnswer As New TextBox

Friend WithEvents btnNext As New Button

Private Sub CreateControls()

lblWelcomeBack.AutoSize = True

lblWelcomeBack.Font = New Font("Microsoft Sans Serif", 11.25!, FontStyle.Regular)

lblWelcomeBack.Location = New Point(25, 12)

lblWelcomeBack.Text = "Welome back, "

Me.Controls.Add(lblWelcomeBack)

btnLogout.Font = New Font("Microsoft Sans Serif", 9.75!, FontStyle.Regular)

btnLogout.Image = My.Resources.logout\_img

btnLogout.ImageAlign = ContentAlignment.MiddleLeft

btnLogout.Location = New Point(834, 12)

btnLogout.Size = New Size(86, 35)

btnLogout.Text = "Log out"

btnLogout.TextAlign = ContentAlignment.MiddleRight

Me.Controls.Add(btnLogout)

lblHomework.AutoSize = True

lblHomework.Location = New Point(26, 45)

lblHomework.Text = "{due tests here}"

Me.Controls.Add(lblHomework)

Me.gpQuizQuestion.Location = New Point(28, 86)

Me.gpQuizQuestion.Size = New Size(889, 439)

Me.gpQuizQuestion.Text = "Question 1"

Me.gpQuizQuestion.Visible = False

Me.Controls.Add(gpQuizQuestion)

lblQuestion.AutoSize = True

lblQuestion.Font = New Font("Microsoft Sans Serif", 11.25!, FontStyle.Regular)

lblQuestion.Location = New Point(348, 18)

lblQuestion.Text = "-Question Here-"

picQuestion.Location = New Point(6, 33)

picQuestion.Size = New Size(324, 316)

picQuestion.SizeMode = PictureBoxSizeMode.StretchImage

txtAnswer.Font = New Font("Microsoft Sans Serif", 12.0!, FontStyle.Regular)

txtAnswer.Location = New Point(351, 285)

txtAnswer.Multiline = True

txtAnswer.Size = New Size(509, 64)

rdoCorrectAnswer.AutoSize = True

rdoCorrectAnswer.Location = New Point(351, 97)

rdoFalseAnswer1.AutoSize = True

rdoFalseAnswer1.Location = New Point(351, 147)

rdoFalseAnswer2.AutoSize = True

rdoFalseAnswer2.Location = New Point(351, 197)

rdoFalseAnswer3.AutoSize = True

rdoFalseAnswer3.Location = New Point(351, 248)

btnNext.Location = New Point(764, 355)

btnNext.Size = New Size(96, 38)

btnNext.Text = "Next"

Dim ctrlCollection As New Collection From {btnNext, lblQuestion, txtAnswer, rdoFalseAnswer3, rdoFalseAnswer2, rdoFalseAnswer1, rdoCorrectAnswer, picQuestion}

For Each ctrl As Control In ctrlCollection

Me.gpQuizQuestion.Controls.Add(ctrl)

Next

End Sub

Sub New()

' This call is required by the designer.

InitializeComponent()

' Add any initialization after the InitializeComponent() call.

CreateControls()

End Sub

#End Region

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

Student.PromptLogout()

End Sub

Private Sub frmStudent\_Closing(sender As Object, e As EventArgs) Handles MyBase.FormClosing

Application.Exit()

End Sub

Private Sub frmStudent\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

ShowControls(False)

lblWelcomeBack.Text = String.Format("Welcome back, {0} {1}.", MyUser.Firstname, MyUser.Lastname)

GetUncompletedQuizzez()

End Sub

Private QuestionsIdQueue As New Queue 'Stores all the questions IDs of the current Quiz.

Private TotalQuestions As Integer 'How many question were initially in the QuestionsID Queue

Private CurrentQuestion As New Question 'This is the currentQuestion that is currently being answered.

Private CurrentQuizID As Integer 'The QuizID of the Quiz the student is currently taking.

Private CurrentScore As Integer 'The students current score.

Private Sub GetUncompletedQuizzez()

QuestionsIdQueue.Clear()

Dim quizList As List(Of Quiz) = Student.GetUncompletedQuizzes(MyUser.UserID)

'Gets a list/array of all the quizzes the student hasn't yet completed.

UpdateLabel(quizList.Count)

CreateQuizLbls(quizList)

End Sub

Private Sub UpdateLabel(count As Integer)

lblHomework.Cursor = Cursors.Default

'Updates the welcome label that tells them if they have any uncompleted quizzes.

'If they do, show them, else let them practise random questions.

RemoveHandler lblHomework.Click, AddressOf PromptRandomQuestions

If count = 0 Then

lblHomework.Text = "You have no outstanding tasks. Click me to practise random set of questions!" \_

& Environment.NewLine & "Note: Your result will not be saved. This is for practising questions."

lblHomework.Cursor = Cursors.Hand

AddHandler lblHomework.Click, AddressOf PromptRandomQuestions

Else

lblHomework.Text = String.Format("You have {0} outstanding homework(s) {1} Click on one to complete it", count, Environment.NewLine)

End If

End Sub

Private Sub ClearLinkLbls()

For i = 1 To Me.Controls.Count

For Each ctrl As CustomLinkLabel In Me.Controls.OfType(Of CustomLinkLabel)()

Me.Controls.Remove(ctrl)

Next

Next

'Clears all the link labels in the form.

End Sub

Private Sub CreateQuizLbls(quizList As List(Of Quiz))

ClearLinkLbls() 'Clears pre-existing CustomLinkLabels on the form

'Initial location

Dim xLocation As Integer = 25

Dim yLocation As Integer = 86

Dim amountCreated As Integer = 0

For i = 0 To quizList.Count - 1

Dim linkLbl As New CustomLinkLabel 'Creates a new instance on each iteration

linkLbl.QuizID = quizList(i).QuizID

linkLbl.Text = quizList(i).Title

linkLbl.Location = New Point(xLocation, yLocation)

linkLbl.AutoSize = True

'Defines the properties of the CustomLinkLabel

amountCreated += 1

yLocation += 33

'Checks the amount of labels created. If it's equal to 10

'It increments the xLocation to move to the right of the screen.

'And resets the yLocation and amount created.

If amountCreated = 10 Then

xLocation += 350

yLocation = 86

amountCreated = 0

End If

'Adds an event handler for the clicked event

AddHandler linkLbl.Click, AddressOf PromptStartQuiz

Me.Controls.Add(linkLbl)

Next

End Sub

Public Sub PromptStartQuiz(sender As Object, e As EventArgs)

'This event is raised when any f the link labels are clicked

Dim quiz = DirectCast(sender, CustomLinkLabel)

'Acceses the quiz the label represents.

If MessageBox.Show("Are you sure you want to start " & quiz.Text & " ?", "Prompt", MessageBoxButtons.YesNo) = DialogResult.Yes Then

QuestionsIdQueue.Clear()

Dim questionIds As List(Of Integer) = Student.GetQuestionsIds(quiz.QuizID)

'Gets a list of the questionIds in the quiz.

For Each QID In questionIds

QuestionsIdQueue.Enqueue(QID)

Next

'Adds each questionID to the QuestionsQueue

TotalQuestions = QuestionsIdQueue.Count

CurrentQuizID = quiz.QuizID

'Set the total questions

'set the current quiz id

CurrentScore = 0

'Resets score and shoes the quiz taking GUI

ShowControls(True)

GetNextQuestionHandler()

'Gets the next question

End If

End Sub

Public Sub PromptRandomQuestions(sender As Object, e As EventArgs)

If MessageBox.Show("Are you sure you want to start?", "Prompt", MessageBoxButtons.YesNo) = DialogResult.Yes Then

CurrentQuizID = Nothing

'Sets current quiz Id to nothing because the student is practising random questions.

Dim questionIds As List(Of Integer) = Student.GetRandomQuestionsIds(New Random().Next(10, 20))

'Gets a random list of questionIds

QuestionsIdQueue.Clear()

For Each QID In questionIds

QuestionsIdQueue.Enqueue(QID)

Next

'Clears the question Ids queue, and enques all the question Ids that it fetched

TotalQuestions = QuestionsIdQueue.Count

UpdateInformation()

ShowControls(True)

GetNextQuestionHandler()

End If

End Sub

Private Sub ShowControls(value As Boolean)

gpQuizQuestion.Visible = value

lblHomework.Visible = Not value

End Sub

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

'When the user answers a question, they click this next button.

'Which marks the question. And gets the next question.

If Student.AnswerIsCorrect(CurrentQuestion.Answer, GetUserAnswer()) Then

CurrentScore += 1

MessageBox.Show("Correct answer")

Else

MessageBox.Show("Incorrect answer")

End If

ClearUserAnswers() 'If they selected any item on the radio buttons

'it unselectes them and clears the textbox they enetered their answer on.

If QuestionsIdQueue.Count = 0 Then

'If 0 then itMeans the queue is empty, so that means student has answered all the questions.

ShowResult(CurrentScore, TotalQuestions)

If CurrentQuizID <> Nothing Then

'If the quizId is nothing it means they are only practising random questions and not a quiz

'So therefore we do not need to save their result.

If Student.SaveQuizResults(CurrentQuizID, MyUser.UserID, CurrentScore, TotalQuestions) = True Then

MessageBox.Show("Your results has been saved successfully!")

Else

MessageBox.Show("There has been an error in trying to save your results!")

End If

'Saves the result, and if succesfull show a message.

End If

CurrentScore = 0 'Resets score

ShowControls(False)

GetUncompletedQuizzez() 'Gets all the uncompleted quizzes

Else

GetNextQuestionHandler()

End If

End Sub

Private Sub GetNextQuestionHandler()

'Gets the next question Id from the QuestionQueue by dequeuing it to get the next question Id.

If QuestionsIdQueue.Count > 0 Then

Dim questionId As Integer = CInt(QuestionsIdQueue.Dequeue)

Dim question As Question = QuestionHelper.GetQuestion(questionId)

lblQuestion.Text = question.QuestionString

picQuestion.Image = Image.FromStream(question.Image.ToStream())

If question.IsMultiChoice Then

question.FalseAnswers(3) = question.Answer

RandomizeAnswers(question.FalseAnswers)

rdoFalseAnswer1.Text = question.FalseAnswers(0)

rdoFalseAnswer2.Text = question.FalseAnswers(1)

rdoFalseAnswer3.Text = question.FalseAnswers(2)

rdoCorrectAnswer.Text = question.FalseAnswers(3)

ChangeControlsVisibility(True)

Else

ChangeControlsVisibility(False)

End If

CurrentQuestion = question

UpdateInformation()

Else

ShowControls(False)

MessageBox.Show("Sorry there are currently no questions avaiable")

End If

End Sub

Private Sub RandomizeAnswers(ByRef qstnArray() As String)

'Takes in a string array.

'Randomizes each item in the array.

For i = 0 To qstnArray.Count - 1

Dim tmp As String = qstnArray(i)

Dim indx As Integer = New Random().Next(qstnArray.Count - i) + i

qstnArray(i) = qstnArray(indx)

qstnArray(indx) = tmp

Next

End Sub

Private Function GetUserAnswer() As String

If CurrentQuestion.IsMultiChoice = False Then

Return txtAnswer.Text

Else

For Each rdo As RadioButton In gpQuizQuestion.Controls.OfType(Of RadioButton)()

If rdo.Checked Then Return rdo.Text

Next

End If

Return String.Empty 'It will never run this line because one of the conditions above must be true

End Function

Private Sub ClearUserAnswers()

'Clears the textbox text and radio buttons checked status so it will be empty for the next question

txtAnswer.Text = String.Empty

For Each rdo As RadioButton In gpQuizQuestion.Controls.OfType(Of RadioButton)()

rdo.Checked = False

Next

End Sub

Private Sub ChangeControlsVisibility(value As Boolean)

rdoCorrectAnswer.Visible = value

rdoFalseAnswer1.Visible = value

rdoFalseAnswer2.Visible = value

rdoFalseAnswer3.Visible = value

txtAnswer.Visible = Not value

End Sub

Private Sub ShowResult(mark As Integer, total As Integer)

Dim percentage As Integer = CInt((mark / total) \* 100)

Dim grade As Char

Dim msg As String = String.Empty

'Translating the percentage into a grade.

'The c next to each string tells the compiler that

'the string is infact a char variable.

Select Case percentage

Case Is <= 39

grade = ("U"c)

Case Is <= 49

grade = ("E"c)

Case Is <= 59

grade = ("D"c)

Case Is <= 69

grade = ("C"c)

Case Is <= 79

grade = ("B"c)

Case Is >= 80

grade = CChar("A")

End Select

If percentage >= 60 Then

msg = "Well done!"

Else

msg = "You can do better!"

End If

Dim text As String = String.Format("{0} you scored {1} out of {2} ({3}%) which is a {4} grade.{5}{6}", MyUser.Firstname, mark, total, percentage, grade, Environment.NewLine, msg)

MessageBox.Show(text)

End Sub

Public Sub UpdateInformation()

gpQuizQuestion.Text = String.Format("Question {0}/{1}", TotalQuestions - QuestionsIdQueue.Count, TotalQuestions)

End Sub

Private Sub txtAnswer\_TextChanged(sender As Object, e As EventArgs) Handles txtAnswer.TextChanged

'A dictionary object is created which contains 2 items per item.

'If the user types in the value on the left, it will replace

' it with the value on the right.

Dim unicode As New Dictionary(Of String, String) From {{"^2", "² "}, {"^3", "³ "}, {"1/2", "½ "}, {"3/4", "¾ "}, {"1/4", "¼ "}}

For Each x As String In unicode.Keys

If txtAnswer.Text.Contains(x) Then

txtAnswer.Text = txtAnswer.Text.Replace(x, unicode(x))

txtAnswer.Select(txtAnswer.Text.Length, 0)

End If

Next

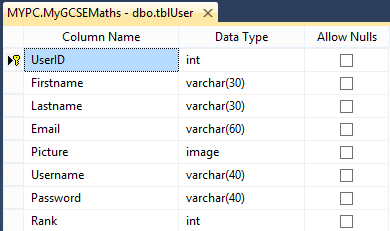
End Sub

End Class

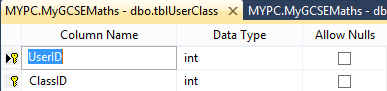
SQL Tables Design view

|  |  |
| --- | --- |
| **Description:** | Design view of all my SQL tables in MSSSQL. |

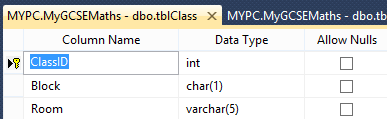
tblUser:



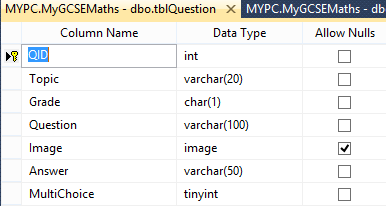
tblUserClass:



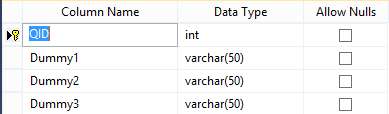
tblClass:



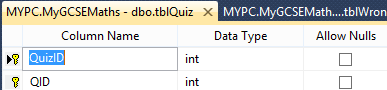
tblQuestion:



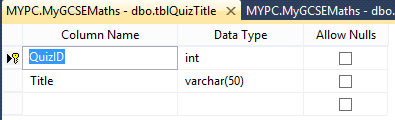
tblWrongAnswers:



tblQuiz:



tblQuizTitle



tblQuizLog:

