TU2983: Advanced Databases

Lab Notes 4:

Inserting New Database Data Through Visual Basic .NET

Written by:

Hafiz Mohd Sarim

Centre of Artificial Intelligence Technology
Fakulti Teknologi dan Sains Maklumat
Universiti Kebangsaan Malaysia

Table of Contents

PART 1: Introduction

A. Lab Objectives

1. This TU2983 lab module aims to fulfil the following objectives:

   i. To teach students how to use SQL INSERT to add new data into database tables from Visual Basic .NET forms.

   ii. To teach students how to capture and display program exceptions (i.e. error messages) without terminating the program.

2. The programming language used for the TU2983 lab modules is Visual Basic .NET, while database used to hold project data will be either of the following
   a. Microsoft Access 2007 or above as the portable database
   b. IBM DB2 Enterprise Server Edition v9.7 or above as the server database.

   Students will be instructed on the exact database to use for each lab exercise and lab assignment.

3. These lab notes were written using the "**Visual Studio 2019 Enterprise**" integrated development environment (IDE) for the **Visual Basic .NET** programming language. However, the steps demonstrated in these lab notes can also be used for all versions of Visual Basic .NET in Visual Studio IDEs starting from Visual Studio .NET 2002 and above.

B. Student Requirements for this Lab

1. All students must bring the following equipment for the lab:

   - **MANDATORY REQUIREMENT:** One USB hard drive / thumb drive pen drive (to save your work)

2. All students must have the following requisite knowledge:

   - **MANDATORY REQUIREMENT:** Using the Microsoft Windows operating system.

   - **MANDATORY REQUIREMENT:** Understand the fundamentals of programming and recognize the different control statements and data structures that are generally available in all programming languages.

   - **MANDATORY REQUIREMENT:** Writing Structured Query Language code, or SQL.

   - **MANDATORY REQUIREMENT:** Creating a sample database in Microsoft Access (see TU2983 Lab Notes 1).

   - **MANDATORY REQUIREMENT:** Building windows forms in Visual Basic .NET and connecting to a Microsoft Access database (see TU2983 Lab Notes 2).

PART 2: Data Manipulation Fundamentals

A. Opening an Existing VB project

1. Start Visual Studio. In the Windows desktop, click **'Start' > 'Microsoft Visual Studio 2019'**.

2. We will use the Visual Basic .NET project you created in TU2983 Lab Notes 2. In the Visual Studio start page, click on **'prj_facultyrecords_a123456'**, which is listed under the **'Recent Projects'** list. Your project will open.

3. Alternatively, you can also browse to the project folder contained in your working folder, for example **'C:\a123456_facultyrecords\prj_facultyrecords_a123456'**, and double-click on the ***.sln** file (solution file), for example **'prj_facultyrecords_a123456.sln'**.

4. At minimum, verify that you have the following forms and modules:
   - A main menu form (**frm_mainmenu_a123456.vb**) [Lab Notes 2]
   - A module containing your global variables (**mod_globals_a123456**) [Lab Notes 3]

5. If you do not have this form and module, please complete TU2983 Lab Notes 2 and 3 before proceeding any further.

B. General Data Access Structure for Writing or Changing Data in a Database

1. The minimum data structures needed to write or change data contained in a database are as follows:

   - **The connection string in the OleDbConnection data type**: This is a string that is specifically formatted to access a particular type of database or read a specific type of database file. This connection string must further also be in the **OleDbConnection** data type [*Also applicable for any other programming language, however data type is only applicable for VB*]

   - **The SQL statement**: This is the Structured Query Language (SQL) operation that you will execute to insert, update, or delete data from the database. [*Also applicable for any other programming language*]

   - **The command or Writer object**: This is the core data structure that allows connectivity to the database file for the purpose of writing or changing data. [*Dependent on the programming language*]

   - **An OPEN command**: This is the command that will open a database connection for writing. [*Dependent on the programming language*]

   - **An EXECUTE command**: This is the command that will use the Writer object to execute the SQL statement. [*Dependent on the programming language*]

   - **A CLOSE command**: This is the command that will close the existing database connection. [*Dependent on the programming language*]

C. Casting your Connection String as an OLEDB Connection Data Type.

1. In TU2983 Lab Notes 3, the connection string to your database is currently stored as a global variable in the **Globals** module, **mod_globals_a123456.vb**, as follows:

```
Module mod_globals_a123456

        Public myconnection as String = "Provider=Microsoft.ACE.OLEDB.12.0;Data
        Source=DB_FACULTYRECORDS_A123456.accdb;Persist Security Info=False;"

End Module
```

2. In its current form, ‘**myconnections’** can remain as a String data type because the **OleDbDataAdapter** method you use to connect to the database to read data using SQL SELECT can accept a String data-typed connection string, as one of its optional parameters.

3. In order to manipulate data in a database through Visual Basic .NET (i.e. inserting, updating, and deleting data) we will be using the **OleDbCommand** method instead.

4. However, the **OleDbCommand** method cannot accept the connection string as a String data type. This method can only recognize connection strings that are formatted in the ‘**OleDbConnection** ‘data type.

5. Therefore you will need to ‘cast’ the existing ‘myconnections’ connection string into a new global variable that has the ‘**OleDbConnection** ‘data type.

> **INFO: ‘Casting’**, also known as ‘cast’ or ‘re-casting’ is a programming practice of changing the data type of an existing variable’s data into a new data type.

6. To cast ‘**myconnections’** to the ‘**OleDbConnection** ‘data type, add the following line to your **Globals** module:

```
Module mod_globals_a123456

        Public myconnection as String = "Provider=Microsoft.ACE.OLEDB.12.0;Data
        Source=DB_FACULTYRECORDS_A123456.accdb;Persist Security Info=False;"

        Public myconnection2 as New OleDB.OleDbConnection(myconnection)

End Module
```

7. You now have the connection string in two data types.

    - You can use ‘myconnection’ when you want to read data with the **OleDbDataAdapter** method,
    - You can use ‘myconnection2’ when you want to write or change data with the **OleDbCommand** method.

PART 3: Adding Database Data

A. Creating a Data Entry Form with SQL INSERT

1. In the Solution Explorer list, add a new form called '**frm_insertcourses_a123456.vb**'.

2. Double-click on the 'My Project' item in the Solution Explorer list and change your **'Startup form'** to '**frm_insertcourses _a123456.vb**'.

3. From your toolbox, add the following objects to the form:

    - On the top of the form, add a **Label**. Change its **(Name)** property to **'lbl_title'**. Change the **'Text'** property of this Label to **'Add New Course'**, and redesign the font of this label as you see fit.

    - Underneath **'lbl_title'**, add a **DataGridView** object. Change its **(Name)** property to **'grd_courses'**. Expand the size of this grid such that it fills two-thirds (2/3) of the form.

    - Underneath **'grd_courses'**, add four **TextBox** objects in a single row.
        - Change the **(Name)** property the of the first **TextBox** to **'txt_code'**.
        - Change the **(Name)** property the of the second **TextBox** to **'txt_name'**.
        - Change the **(Name)** property the of the third **TextBox** to **'txt_credit'**.
        - Change the **(Name)** property the of the fourth **TextBox** to **'txt_dept'**.
        - Add additional Label objects to label each textbox accordingly.

    - Underneath all the textboxes, add a **Button** object. Change its **(Name)** property to **'btn_insert'**. Change the **'Text'** property of this Button to **'INSERT'.**

4. Open the **Form_Load** event for this form, by double-clicking on any blank space in the '**frm_insertcourses_a123456.vb**' form.

5. Underneath the '**End Sub**' of the **Form_Load** event, write the following to create a sub-procedure called '**refresh_grid()**':

```
Private Sub refresh_grid()

        Dim mysql as String = "SELECT * FROM TBL_COURSES_A123456"

        Dim mydatatable as New DataTable

        Dim myreader as New OleDb.OleDbDataAdapter(mysql, myconnection)

        myreader.Fill(mydatatable)

        grd_courses.DataSource = mydatatable

End Sub
```

6. Underneath the '**End Sub**' of the '**refresh_grid()**' function call, write the following to create a sub-procedure called '**clear_fields()**':

```
Private Sub clear_fields()

        txt_code.Text = ""
        txt_name.Text = ""
        txt_credit.Text = ""
        txt_dept.Text = ""

End Sub
```

7. In the **Form_Load** event add the '**refresh_grid()**' function call:

```
Private Sub frm_insertcourses_a123456_Load(sender As System.Object, e as System.EventArgs)
Handles MyBase.Load

        refresh_grid()

End Sub
```

8. Double-click on btn_insert to generate its **'Button_Click'** event code snippet. Add the following lines of code into the '**Button_Click'** event:

```
Private Sub btn_insert_Click(sender As System.Object, e As System.EventArgs) Handles
btn_insert.Click

        Dim mysql As String = "INSERT INTO TBL_COURSES_A123456 VALUES ('" & txt_code.Text
        & "', '" & txt_name.Text & "', " & txt_credit.Text & ", '" & txt_dept.Text & "')"

        Dim mywriter As New OleDb.OleDbCommand(mysql, myconnection2)

        mywriter.Connection.Open()
        mywriter.ExecuteNonQuery()
        mywriter.Connection.Close()

        refresh_grid()
        clear_fields()

End Sub
```

9. Note that for **txt_credit.Text** there are no enclosing quote marks (' ') before or after this object property reference. This is because the '**fld_credits'** attribute that it corresponds to in the database tables is of a numeric (i.e. **Integer**) data type.

10. Run the program, and insert the following data into the TextBoxes:

    - **Course code = TU3263**
    - **Course name = Computer Security**
    - **Credits = 3**
    - **Managing Department = TU**

11. Save your project.

B. Forcing a Programming Exception

1. We can evaluate the resilience of our Visual Basic code by attempting to cause the program to 'throw' (i.e. output) an exception (or error) during normal operation.

2. Data entry mistakes by end-users can cause your program to terminate due to an error, even though it has been programmed correctly.

3. Examples of mistakes that end-users can inadvertently cause during data entry are as follows:

   - Leaving the primary key field blank when inserting a new row of data
   - Entering an existing primary key value when inserting a new row of data
   - Entering an invalid foreign key value for foreign key fields
   - Entering String data (text) into a numeric field

4. Run the '**frm_insertcourses_a123456.vb**' form again. This time enter the following data into the TextBoxes:

   - **Course code = TU3263**
   - **Course name = Final Year Project**
   - **Credits = 12**
   - **Managing Department = TU**

5. You will notice your program will terminate to debug mode with the following error message

"The changes you requested to the table were not successful because they would create duplicate values in the index, primary key, or relationship. Change the data in the field or fields that contain duplicate data, remove the index, or redefine the index to permit duplicate entries and try again."

6. More importantly your program terminated in such a way that the end user cannot correct his mistake.

7. Preferably, we would like the program to go back to the state before the error occurred, without terminating, so that the user can correct his mistake.

C. Catching errors

1. Edit your ‘Button_click’ code snippet to include the following lines:

```
Private Sub btn_insert_Click(sender As System.Object, e As System.EventArgs) Handles
btn_insert.Click

        Dim mysql As String = "INSERT INTO TBL_COURSES_A123456 VALUES (‘” & txt_code.Text & "’,
        ‘" & txt_name.Text & "’, " & txt_credit.Text & ", ‘" & txt_dept.Text & "’)"

        Dim mywriter As New OleDb.OleDbCommand(mysql, myconnection2)

        Try

                mywriter.Connection.Open()
                mywriter.ExecuteNonQuery()
                mywriter.Connection.Close()

                refresh_grid()
                clear_fields()

        Catch ex As Exception

                Beep()
                MsgBox(“There is a mistake in the data you entered, as shown below” & vbCrLf
                & vbCrLf & ex.Message)

                mywriter.Connection.Close()

        End Try

End Sub
```

2. Run the ‘**frm_insertcourses_a123456.vb**’ form again. Once again enter the following data into the TextBoxes:

   - **Course code = TU3263**
   - **Course name = Final Year Project**
   - **Credits = 12**
   - **Managing Department = TU**

3. Note that this time the error message still appears but the program does not terminate.

4. This **Try...Catch** statement would attempt to execute the code between the ‘**Try’** and ‘**Catch’** lines. If it faces an error, the program will not run the remainder of the code in the ‘**Try’** portion. They are skipped over, and the code in the ‘**Catch’** portion is executed.

5. In this case, it will produce an audible beep. capture the error message, display the error message in a message box, and close the database connection (because the database was ‘opened’ before the error occurred in the **ExecuteNonQuery** method.

6. The end-user can resume changing the data, before trying to insert again. Now, correct the mistake in the **txt_code.Text** as follows:

   - **Course code = TU3983**

7. Click the INSERT button again. This time the data entry is successful.

PART 4: Creating Functions

A. Creating a General Purpose SQL Query Execution Function

1. You may notice that the VB code for executing an SQL query, such as the one used in the **refresh_grid()** sub-procedure earlier is almost the same as all the SQL query code that you written so far.

2. The code snippet of the **refresh_grid()** sub-procedure below, highlights which lines of code are the same and which lines are different:

```
Private Sub refresh_grid()

        Dim mysql as String = "SELECT * FROM TBL_COURSES_A123456"    '<-- DIFFERENT

        Dim mydatatable as New DataTable   '<-- SAME

        Dim myreader as New OleDb.OleDbDataAdapter(mysql, myconnection)   '<-- SAME

        myreader.Fill(mydatatable)   '<-- SAME

        grd_courses.DataSource = mydatatable   '<-- DIFFERENT

End Sub
```

3. The lines that are different are specific to the form where the **refresh_grid()** sub-procedure is executed. Therefore, these two lines will need to be changed whenever we want to execute a query on any other form. Furthermore, these changes forces us to recreate the **refresh_grid()** sub-procedure, and tailor it to suit for every form.

4. A more efficient way of doing this is declare your SQL code as a **global function** in your **'mod_globals_a123456'** module. This global function will serve as a general purpose SQL query execution function that can be used anytime you need to run an SQL SELECT event.

5. In VB.net, **'sub-procedures'** and **'functions'** are both chunks of code that can be called using its sub-procedure name or function name. However, functions are different from sub-procedures because all functions:
   i. need to be declared with a data type
   ii. must return a value

   ... while sub-procedures do not need either of these.

6. Add the following code to your **'mod_globals_a123456'** module to declare this global function:

```
Public Function run_sql_query(mysql As String) As DataTable

        Dim mydatatable As New DataTable
        Dim myreader As New OleDb.OleDbDataAdapter(mysql, myconnection)

        Try
                myreader.Fill(mydatatable)
        Catch ex As Exception

                Beep()
                MsgBox("There is an error:" & vbCrLf & vbCrLf & ex.Message)
        End Try

        Return mydatatable

End Function
```

7. You can now use this **run_sql_query()** function to replace your SQL query execution code. By inputting a SELECT command in the run_sql_query() function, you can use at as a DataTable object.

B. Creating a STUDENT Data Entry Form

1. In your Solution Explorer panel, add a new Form to your **'prj_facultyrecords_a123456'** project, and name your new form **'frm_insertstudent_a123456.vb'**.

2. Add a new Label in this form. Rename the Label as **'lbl_title'**, and change its **'Text'** property to **'Insert New Students'**.

3. Add a **DataGridView** object to this form, and close any context window that appears. Expand the **DataGridView** object to fill almost the entire form. Rename the **DataGridView** object as **'grd_students'**.

4. Double click on any **empty form area**, to generate the **'Form_Load'** event code snippet.

5. In the **'frm_insertstudent _a123456_Load'** code snippet, type in the following:

```
Private Sub frm_insertstudent_a123456_Load(sender As System.Object, e As System.EventArgs)
Handles MyBase.Load

        grd_students.DataSource = run_sql_query("SELECT * FROM TBL_STUDENTS_A123456")

End Sub
```

6. Run your program. When the **'frm_insertstudent _a123456'** form opens, the results of your SQL statement will be displayed in the form's grid.

7. Save your project.

C. Using String Manipulation Functions to Generate New Primary Keys

1. In the **'frm_insertstudent _a123456_Load'** code snippet, add in the following code:

```
Private Sub frm_insertstudent_a123456_Load(sender As System.Object, e As System.EventArgs)
Handles MyBase.Load

        grd_students.DataSource = run_sql_query("SELECT * FROM TBL_STUDENTS_A123456")

        Dim lastmatric As String = run_sql_query("SELECT MAX(FLD_MATRIC) AS LASTMATRIC
        FROM TBL_STUDENTS_A123456").Rows(0).Item("LASTMATRIC")

        MsgBox(lastmatric)

End Sub
```

2. Run your program. Before your form opens, a MessageBox will appear and show you the last matric number currently in your STUDENTS table.

3. In order to add a new student, we will need to give a new unique value for the FLD_MATRIC attribute since it is a primary key in the STUDENTS table.

4. We can use **string manipulation functions** to generate a new matric number value from the **lastmatric** variable.

5. The **Mid()** function reads the characters of a string from a starting character position (mandatory) to an ending character position (optional). We can use the Mid() function to extract the numeric portion of the last matric number, increment it by 1, and re-attach the matric prefix (“A”)

6. Add 3 TextBox objects to your form beneath the **grd_students**, and change their (Name) properties to **‘txt_matric’, ‘txt_name’**, and **‘txt_dept’**.

7. Cut the lines you have just added in step 1, and add it into a new function called '**generate_matric()**' in your form, which you will add just beneath the **'frm_insertstudent _a123456_Load'** code snippet. Modify your code as follows:

```
Private Sub frm_insertstudent_a123456_Load(sender As System.Object, e As System.EventArgs)
Handles MyBase.Load

        grd_students.DataSource = run_sql_query("SELECT * FROM TBL_STUDENTS_A123456")

        txt_matric.Text = generate_matric()

End Sub

Private Function generate_matric() As String

        Dim lastmatric As String = run_sql_query("SELECT MAX(FLD_MATRIC) AS LASTMATRIC
        FROM TBL_STUDENTS_A123456").Rows(0).Item("LASTMATRIC")

        MsgBox(lastmatric)

        Dim newmatric As String = "A" & Mid(lastmatric, 2) + 1

        Return newmatric

  End Function
```

8. Run your program. You will see the last matric number in the message box, while the new matric number will be displayed in your first textbox, '**txt_matric**'.

9. Delete or comment out the existing MessageBox line.

10. Add a Button object at the bottom of your form, and change its (Name) property to **'btn_insert'**.

11. Double-click on **btn_insert** to generate its **'Button_Click'** event code snippet. Add the following lines of code into the '**Button_Click'** event:

```
Private Sub btn_insert_Click(sender As System.Object, e As System.EventArgs) Handles
btn_insert.Click

        Dim mysql as String = "INSERT INTO TBL_STUDENTS_A123456 VALUES ('" &
        txt_matric.Text & "', '" & txt_name.Text & "', '" & txt_dept.Text & "')"

        Dim mywriter As New OleDb.OleDbCommand(mysql, myconnection2)

        Try

                mywriter.Connection.Open()
                mywriter.ExecuteNonQuery()
                mywriter.Connection.Close()

                grd_students.DataSource = run_sql_query("SELECT * FROM
                TBL_STUDENTS_A123456")

                txt_matric.Text = generate_matric()
                txt_name.Text = ""
                txt_dept.Text = ""

        Catch ex As Exception

                Beep()
                MsgBox("There is a mistake in the data you entered, as shown below" & vbCrLf
                & vbCrLf & ex.Message)

                mywriter.Connection.Close()

        End Try

End Sub
```

12. Note that you no longer have to declare another **refresh_grid()** function.

13. Run your program. When the **'frm_insertstudent _a123456'** form opens, add in a new student record using the information listed below, and click the INSERT button:

- **Matric Number = A108** (automatically created for you)
- **Name = Bob**
- **Department = TR**

14. The new student will be displayed in the grid.

15. Save your project.

EXERCISE 1: Create a Data Entry form for the 'Department' table (TBL_DEPARTMENT_A123456).

PART 5: File Selection

A. Adding a Default Picture

1. Close the **'frm_insertstudent _a123456'** form.

2. Double-click on the 'My Project' item in the Solution Explorer list and change your **'Startup form'** to '**frm_studentdetails _a123456.vb**', which you have previously created in LAB NOTES 3, and run the program.

3. In the list of matric numbers shown in **lst_matric**, click in the matric number of the new student that you have just inserted (A108).

4. The program will terminate with the error message "FileNotFoundException was unhandled". This error message appears because when you inserted the new student record in the **'frm_insertstudent _a123456'** form, this new student does not yet have a picture.

5. You can add a temporary picture as a 'Default' picture to display for any records that do not have pictures.

6. In your web browser, open http://images.google.com and Google "no photo". Select a suitable picture that indicates that no picture is available, and save it in the **'pictures'** subfolder as **'nophoto.jpg'**.

7. Modify the 'reftesh_text()' sub-procedure as follows:

```
Private Sub refresh_text(matric As String)

        Dim mysql As String = "SELECT * FROM TBL_STUDENTS_A123456 WHERE FLD_MATRIC='" & matric &
        "'"

        Dim mydatatable As New DataTable

        Dim myreader As New OleDb.OleDbDataAdapter(mysql, myconnection)

        myreader.Fill(mydatatable)

        txt_matric.Text = mydatatable.Rows(0).Item("FLD_MATRIC")
        txt_name.Text = mydatatable.Rows(0).Item("FLD_NAME")
        txt_dept.Text = mydatatable.Rows(0).Item("FLD_DEPT")

        Try
                pic_student.BackgroundImage = Image.FromFile("pictures/" & txt_matric.Text & ".jpg")
        Catch ex As Exception
                pic_student.BackgroundImage = Image.FromFile("pictures/nophoto.jpg")
        End Try

End Sub
```

8. Run your program again.

9. In the '**frm_studentdetails _a123456.vb**' form, once again click in the matric number of the new student that you have just inserted (A108). This time, the program does not terminate, but instead shows the default picture for students with no photos.

B. Using a Default Picture File for Data Entry

1. To avoid potential errors that may occur due to missing associated files (such as photos) for database records, these associated files should be uploaded at the point of data entry. In other words, the associated file should be uploaded in the same time that the new database record is entered. In our case, we would like to add a new student's photo at the same time that the new student's record is inserted into the database.

2. Close the **'frm_studentdetails _a123456'** form.

3. Double-click on the 'My Project' item in the Solution Explorer list and change your **'Startup form'** to '**frm_ insertstudent _a123456.vb**', which you have previously created in **Part 4.B** of these notes, and view the code for the form.

4. Declare the following form variable at the top of the form's code. This variable shows the current location of our default picture:

```
Public Class frm_insertstudent_a123456

        Dim defaultpicture As String = Application.StartupPath & "\pictures\nophoto.jpg"
```

5. Add a new TextBox object, and a new PictureBox object to the form. Modify these new object properties as follows:

   - TextBox: (Name) = **txt_picture**
   - PictureBox: (Name) = **pic_student** , BackgroundImageLayout = **Zoom**

6. Modify the 'Form_Load' event (frm_insertstudent_a123456_Load), by adding the following lines. This will display the default picture, and the path (i.e. file location) of the default picture when the form in first opened:

```
Private Sub frm_insertstudent_a123456_Load(sender As System.Object, e As System.EventArgs)
Handles MyBase.Load

        grd_students.DataSource = run_sql_query("SELECT * FROM TBL_STUDENTS_A123456")

        txt_matric.Text = generate_matric()

        txt_picture.Text = defaultpicture
        pic_student.BackgroundImage = Image.FromFile(defaultpicture)

End Sub
```

7. Finally, modify the code for the ‘btn_insert_Click’ event by adding the following lines:

```
Private Sub btn_insert_Click(sender As System.Object, e As System.EventArgs) Handles
btn_insert.Click

        Dim mysql as String = "INSERT INTO TBL_STUDENTS_A123456 VALUES ('" & txt_matric.Text &
        "', '" & txt_name.Text & "', '" & txt_dept.Text & "')"

        Dim mywriter As New OleDb.OleDbCommand(mysql, myconnection2)

        Try

                mywriter.Connection.Open()
                mywriter.ExecuteNonQuery()
                mywriter.Connection.Close()

                My.Computer.FileSystem.CopyFile(txt_picture.Text, "pictures\" &
                txt_matric.Text & ".jpg")

                grd_students.DataSource = run_sql_query("SELECT * FROM TBL_STUDENTS_A123456")

                txt_matric.Text = generate_matric()
                txt_name.Text = ""
                txt_dept.Text = ""
                txt_picture.Text = defaultpicture
                pic_student.BackgroundImage = Image.FromFile(defaultpicture)

        Catch ex As Exception

                Beep()
                MsgBox("There is a mistake in the data you entered, as shown below" & vbCrLf  &
                vbCrLf & ex.Message)

                mywriter.Connection.Close()

        End Try

End Sub
```

8. This will make a copy of the current default picture **“nophoto.jpg”** into a new file that matches the student’s matric number as shown in **txt_matric.Text**, and it will place this copy in the **‘pictures’** subfolder.

C. Selecting a Picture File from System Folders

1. Ideally, the user should be able to select a picture that the user wants for the new student record to be entered. For this example we will assume that the user has saved a new student’s picture in a temporary location, such as the Windows Desktop.

2. In your web browser, open http://images.google.com and Google “passport photo”. Copy a suitable passport photo of a person, and save it on the Windows Desktop. You may leave the name of the photo unchanged.

3. Add a new Button object, and an **OpenFileDialog** object to the form. Modify these new object properties as follows:

   - Button: (Name) **= btn_picture** , Text = **‘Select Picture’**
   - OpenFileDialog: (Name) **= OpenFileDialog1** (leave it as its default value)

4. Double-click on **btn_picture** to generate its **'Button_Click'** event code snippet. Add the following lines of code into the '**Button_Click**' event:

```
Private Sub btn_picture_Click(sender As System.Object, e As System.EventArgs) Handles
btn_picture.Click

End Sub
```

5. Declare a String variable that points to the path (i.e. folder location) of the Windows Desktop, by adding this code:

```
Private Sub btn_picture_Click(sender As System.Object, e As System.EventArgs) Handles
btn_picture.Click

        Dim mydesktop As String = My.Computer.FileSystem.SpecialDirectories.Desktop

End Sub
```

6. Next, we will use the OpenFileDialog object, OpenFileDialog1, to select the picture file, by adding the following code. This will pop up a file selection window that starts at the Windows Desktop, and filter the files to show only JPG image files which we will keep as our student photos. The user can select the file he wants from this file selection window:

```
Private Sub btn_picture_Click(sender As System.Object, e As System.EventArgs) Handles
btn_picture.Click

        Dim mydesktop As String = My.Computer.FileSystem.SpecialDirectories.Desktop

        OpenFileDialog1.InitialDirectory = mydesktop
        OpenFileDialog1.FileName = ""
        OpenFileDialog1.Filter = "JPG files (*.jpg)|*.jpg"
        OpenFileDialog1.ShowDialog()

End Sub
```

7. Finally, add the following code to display the picture from the user selected file into the **pic_student** PictureBox object, and to record the location of the user selected file into **txt_picture.Text**. This new location will now point to the file that the user wants to copy to the '**pictures**' subfolder for this student's record:

```
Private Sub btn_picture_Click(sender As System.Object, e As System.EventArgs) Handles
btn_picture.Click

        Dim mydesktop As String = My.Computer.FileSystem.SpecialDirectories.Desktop

        OpenFileDialog1.InitialDirectory = mydesktop
        OpenFileDialog1.FileName = ""
        OpenFileDialog1.Filter = "JPG files (*.jpg)|*.jpg"
        OpenFileDialog1.ShowDialog()

        pic_student.BackgroundImage = Image.FromFile(OpenFileDialog1.FileName)
        txt_picture.Text = OpenFileDialog1.FileName

End Sub
```

8. Save your program.

9. Test your program by running the **'frm_insertstudent _a123456'** form, and add in a new student record using the information listed below, but do not click the INSERT button yet:

   - **Matric Number = A109** (automatically created for you)
   - **Name = Nora**
   - **Department = TK**

10. Click the 'Select Picture' button, and when the file selection dialog appears select the picture that you have previously saved to the desktop (see Part 5.C.2) and click '**Open**'.

11. When the file selection dialog closes, verify that the correct picture is displayed in the picture box, and the correct file path is shown in the **txt_picture** TextBox object.

12. Finally, click the INSERT button to save the new student record to the database, and to copy the new student's picture to the 'pictures' subfolder at the same time.

13. Verify that your new record and picture has been saved correctly by running the **'frm_studentdetails _a123456'** form, and selecting the new student's matric number from the ListBox object.

PART 6: Before Leaving the Lab

1. Copy your entire working folder, e.g **'c:\a123456_facultyrecords'**, to your USB drive. This folder will be used again for the next lab session.

2. Delete the existing **'c:\a123456_facultyrecords'** folder in your lab computer's **'C:\'** drive.

Additional Lab Exercises

1. Create a new form for inserting new Departments into your database.

2. In your existing frm_insertcourses_a123456 form, add a ComboBox that lets the user select a Department ID, and add a button that displays the largest Course Code in the database for that Department ID in a MsgBox. Assume that the first 2 letters of a Course Code indicates the Department ID that offers the course

3. In your existing frm_insertcourses_a123456 form, add a second ComboBox that lets the user select a 'Student Year' value (1, 2, 3, or 4), and modify the button from Exercise 2 above to now display the largest Course Code in the database for that pair of Department ID and Student Year in a MsgBox. Assume that the first digit of a Course Code indicates the Student Year when the course is taken.

4. In your existing frm_insertcourses_a123456 form, add a third ComboBox that lets the user select a 'Credit Hour' value (3, or 4), and modify the button from Exercise 3 above to now display the largest Course Code in the database for that combination of Department ID, Student Year and Credit Hour in a MsgBox. Assume that the last digit of a Course Code indicates the Credit Hours of the course.

5. In your existing frm_insertcourses_a123456 form, modify the button from Exercise 4 above to now generate a new unique Course Code in the database for that combination of Department ID, Student Year and Credit Hour. This newly generated Course Code must be displayed in the existing TextBox for Course codes (txt_code). This newly generated Course Code must also conform to the structure of existing course codes, where:

   - The first 2 letters indicates the Department ID
   - The first digit indicates the Student Year
   - The last digit indicates the Credit Hours