TSSM™

Creating VCE Success

Software Development Teach Yourself Series

Topic 8: Data Validation and Testing Units 1,3,4

A: Level 14, 474 Flinders Street Melbourne VIC 3000
T: 1300 134 518 **W:** tssm.com.au **E:** info@tssm.com.au

Contents

CASE STUDY

All Teach Yourself Series in this package will refer to the following case study.

Tariq Mulner is the manager of a school canteen. He manages how many lunches are going to be prepared each day. It is difficult to tell how many lunches will be sold, so he would like a software solution that students can use to order lunches. This application would provide him with a complete list of orders.
Most of the students have smart phones, so Tariq is suggesting the solution is a phone app that can read in the lunch order, and send it to his device so he can print out the order list.

Photo by Tirachard Kumtanom from Pexels used with permission

Data Validation

Data Validation is part of the Development Stage of the PSM. It is crucial that all data entered into the Software Solution is validated using a wide range of techniques.

There are three key types of Validation:

1. Existence Checking
2. Data Type Checking
3. Range Checking

1. Existence Checking tests to see if the data exists. For example, if the user has not entered their name data into the form then the form cannot be processed. Validation would alert the user to enter data that has not been entered.

Pseudocode Example

```
Checking a textbox
    IF TextBox.Text <> “” THEN
        Read in TextBox.Text Data
    Else
        Message(“Please enter data”)
    EndIf
```

2. Type Checking tests to see if the data type is appropriate for the processes. Setting a Type Check on a postcode can further limit data entry to values returning a warning if letters or other symbols are entered.

```
Checking a Type
    NumericCheck(Value)
    IF NumericCheck(Value) = True THEN
        Message(“This value is the correct data type”)
    Else
        Message (“Please enter numbers only”)
    EndIf
```

3. Range Checking tests to see if the entered data is within a set range. For example a post code in Australia consists of only 4 values. Postcodes in NSW all start with "2". If NSW is a restriction to the post code then a range check can test for input to be between 2000 and 2999. A common range check exists with date of birth data. This type of range check can check if the user's Date of Birth makes them eligible over 18 or to test if the data is valid. Anyone entering a year over 100 years may be considered out of range.

```
Checking a Range
    IF Value <= 100 AND Value >= 10 THEN
        Message(“This Value is IN RANGE”)
    Else
        Message (“Please enter data between 10 and 100”)
    EndIf
```

When adding a full set of validation on typed in data it should follow this order:

Existence Check → Data Type Check → Range Check

The Canteen Application contains a textbox called txtQty. This is where the user will enter the quantity of items for their lunch order. Tariq has decided that no one can order more than 3 items and cannot leave the text box empty.

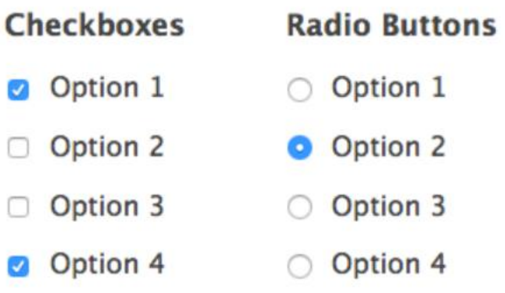
Below is an algorithm to demonstrate how all three validation techniques can be used to fully check the input on the intQty variable.

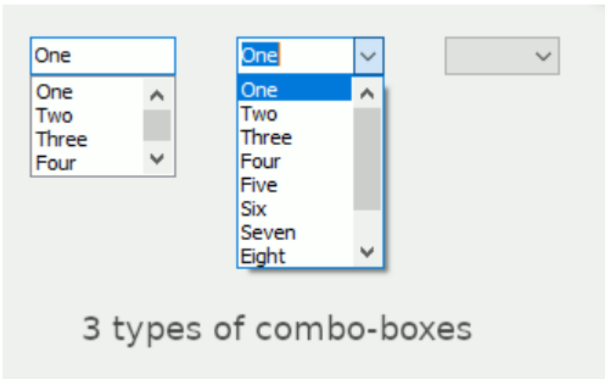
```
Start
    If txtQty.Text <> "" Then
        NumericCheck = IsNumeric(txtQty.Text)
        If NumericCheck = True Then
            If (txtQty.Text > 0) And (txtQty.Text < 4) Then
                intQty = txtQty.Text
            Else
                MsgBox("You have entered a value too large – please enter a value between 1 and 3.")
            End If
        Else
            MsgBox("Please enter a number for your quantity")
        End If
    Else
        MsgBox("Please enter your quantity")
    End If
End
```

Other methods of Validation include the restriction of data input using the following interface objects:

- Radio buttons
- Check Boxes
- ComboBoxes / PullDown Menus
- Input Masks (eg: --/--/---- to indicate a date format)

Radio Buttons: limit selections to only ONE.
Check Boxes: allow for more than one item to be selected.





There are three kinds of ComboBoxes:

1. Simple Combo Box: an edit control is placed on top of a list box. Users can choose from the list ***or*** type text in the edit box. You can use this style for short lists.

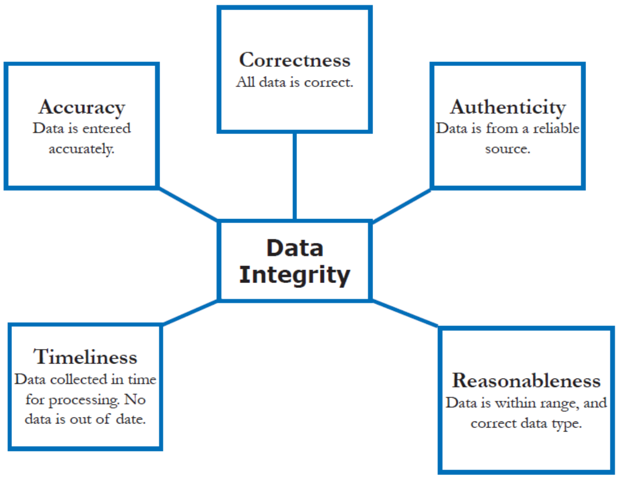
2. Drop-down: similar to the simple type, except that the list box isn't displayed until a drop-down button is clicked. To be used for long lists, or if there is not much room on the form.

3. Drop-down list: just like the drop-down style, the list isn't visible at first. Users can click the drop-down button and chose from the list, but they can't enter text in the edit portion. Use this type when you want to select only from a fixed set of choices.

Data Validation and Verification

There are other kinds of validation required before identifying your test data to run your program. You may also be aware of these type of data validation to apply to the Case Study in U4O2. When any data is collected to be input into a system, for the data to be:

- Correct
- Accurate
- Timely
- Authentic
- Reasonable


Correctness
All data is correct.
Accuracy
Data is entered accurately.
Authenticity
Data is from a reliable source.
Data Integrity
Timeliness
Data collected in time for processing. No data is out of date.
Reasonableness
Data is within range, and correct data type.

Validation type	How it works	Example usage
Check digit	The last one or two digits in a code are used to check the other digits are correct	Bar code readers in supermarkets use check digits
Format check	Checks the data is in the right format	A National Insurance number is in the form LL 99 99 99 L where L is any letter and 9 is any number
Length check	Checks the data isn't too short or too long	A password which needs to be six letters long
Lookup table	Looks up acceptable values in a table	There are only seven possible days of the week
Existence check	Checks that data has been entered into a field	In most databases a key field cannot be left blank
Range check	Checks that a value falls within the specified range	Number of hours worked must be less than 50 and more than 0
Spell check	Looks up words in a dictionary	When word processing

Verification

There are two main methods of verification:

Double entry - entering the data twice and comparing the two copies. This effectively doubles the workload, and as most people are paid by the hour, it costs more too.

Proofreading data - this method involves someone checking the data entered against the original document. This is also time-consuming and costly.

Testing

Testing for errors is a major part of the development stage. There are three main types of errors you will encounter:

1. Syntax Errors
2. Runtime Errors
3. Logic Errors

Syntax Errors

Most commonly syntax errors occur when the programmer is not familiar with the programming language or has made typing errors. Syntax errors are picked up by the compiler or the interpreter (such as a browser) and return with instructions on how to fix them. Programs rarely run with syntax errors especially if they are compiled languages. Interpreted languages such as PHP or JavaScript will run the code up to error.

Runtime Errors

These may be picked up by compilers or interpreters and often result in data not being calculated.

Arithmetic Overflow occurs when a value is being calculated outside of the memory size allocated for the variable handling it. If the variable can only handle 2 bytes and a value of 34972.8776 is produced to be stored in that variable there will not be enough memory to store it and the program will crash.

Logic Errors

Logic errors can only be discovered by running the software solution with a wide range of test data. Choosing test data is the crucial element for finding logic errors. Logic errors have been implemented by the programmer in at the algorithm stage and really should be ironed out before coding commences.

The Canteen Application Algorithm

HotFood(5,2)

	0	**1**	**2**	**3**	**4**
0	Pies	Sausage Rolls	Hot Dogs	Hot Chips	Dim Sims
1	$2.40	$1.50	$2.30	$1.20	$2.00

```
START Type
    On Enter Button:
    Set ComboBox to “Please Select Food Type”
    CASE ComboBox Selected Item
        Case 0: Open Form “HotFood”
        Case 1: Open Form “Sandwiches”
        Case 2: Open Form “Sushi”
    End CASE
    IF CASE = “Please Select Food Type” THEN
        Message(“Please select a Food Type”)
    END IF
END

START HotFood
    On Enter Button:
    Set ComboBox to “Please Select Hot Food Item Lunch Item”
    CASE ComboBox Selected Item
        Case 0: fptPrice←HotFood(0,1)
        Case 1: fptPrice←HotFood(1,1)
        Case 2: fptPrice←HotFood(2,1)
        Case 3: fptPrice←HotFood(3,0)
        Case 4: fptPrice←HotFood(4,1)
    End CASE
    IF CASE = “Please Select Hot Food Item” THEN
        Message(“Please select a Hot Food Item”)
    END IF
    If txtQty.Text <> “” Then
        NumericCheck = IsNumeric(txtQty.Text)
        If NumericCheck = True Then
            If (txtQty.Text =3) Then
                intQty ← txtQty.Text
            Else
                MsgBox(“You must enter 1, 2 or 3 into the Quantity Box”)
            End If
        Else
            MsgBox(“Please enter a value for your quantity”)
        End If
    Else
        MsgBox(“Please enter your quantity”)
    End If

    ftpCost← intQty * fptPrice
    Display ftpCost
End
```

You have written the code to implement the algorithms above. You have tried to get it working with some reasonable data entries, but you are finding a couple of logic error.

Finding Errors

The program only works if the quantity is 3. It doesn't work for 1 or 2 items. The program crashes when you select Hot Chips.

A testing table is a tool used to document the removal of errors from the solution. It needs to investigate each variable and compare the expected result with the actual results from the software in development.

TESTING TABLE

Food Type	Quantity	Expected fptCost	Actual fptCost
Pies	1	2.40	You must enter 1, 2 or 3 into the Quantity Box
Pies	2	4.80	You must enter 1, 2 or 3 into the Quantity Box
Pies	3	7.20	7.20
Hot Chips	1	1.20	Crashes
Hot Chips	2	2.40	Crashes
Hot Chips	3	3.60	Crashes

We need to assemble some test data. The data needs to test the condition used in the IF Statement. As you can see in the testing table above 3 Pies works, but not 2 Pies or 1 Pie. This indicates it is a logic error. Hot Chips does not work at all. The crash indicates a runtime error.

When we encounter a logic error, we need to use a Trace Table. A trace table allows you to find the error by painstakingly moving data through the program line by line to find the error.

TRACE TABLE

FoodType Selected	fptPrice	intQty	fptCost
Pies HotFood(0,1)	$2.40	2	You must enter 1, 2 or 3 into the Quantity Box
Error found in Range Check condition (txtQty.Text =3) It should be (txtQty.Text<>0) AND (txtQty.Text < 4)			
Hot Chips HotFood(3,0)	"Hot Chips"		
Error found in CASE 3: Case 3: fptPrice←HotFood(3,0) It should be replaced with Case 3: fptPrice←HotFood(3,1)			

Test Data

When assembling test data there are key elements to be taken into consideration:

- Valid and invalid data to investigate how robust the solution is to invalid input.
- Data that tests conditions in loops and decision control structures.

Review Questions

Applied Questions

Tariq has asked you to produce an app. The Analysis is available in TOPIC 1. You have 3 weeks to develop the app to hand over. Ideally Tariq should have the solution for 2 weeks before evaluation begins.

1. What object would enhance the validation on the intQty variable?

A. Input Mask

B. ComboBox

C. Radio Button

2. A common problem with the prototype has been students not entering their student number correctly. Often students forget the letter at the front and the dash. What form of validation could be used to ensure students enter their ID correctly?

A. Existence Check

B. Input Mask

C. ComboBox

3. We need to assemble some test data to test our faulty algorithm . The data needs to test the condition used in the IF Statement. As you can see in the testing table above 3 Pies works, but not 2 Pies or 1 Pie. This indicates it is a logic error. Hot Chips does not work at all. The crash indicates a runtime error.

When we encounter a logic error, we need to use a Trace Table. A trace table allows you to find the error by painstakingly moving data through the program line by line to find the error.

TRACE TABLE

FoodType Selected	**fptPrice**	**intQty**	**fptCost**

TSSM™
Creating VCE Success

Solutions to Review Questions

1. B. ComboBox – is the most efficient option and uses less screen space

2. B. Input Mask

3.

TRACE TABLE

FoodType Selected	**fptPrice**	**intQty**	**fptCost**
Pies HotFood(0,1)	$2.40	2	You must enter 1, 2 or 3 into the Quantity Box
Error found in Range Check condition (txtQty.Text =3) It should be (txtQty.Text<>0) AND (txtQty.Text < 4)			
Hot Chips HotFood(3,0)	"Hot Chips"		
Error found in CASE 3: Case 3: fptPrice←HotFood(3,0) It should be replaced with Case 3: fptPrice←HotFood(3,1)			