TSSM™
Creating VCE Success

Software Development Teach Yourself Series

Topic 5: Design Tools Units 1,3

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

Design Tools

In the Software Development Study Design, we focus on the following design tools:

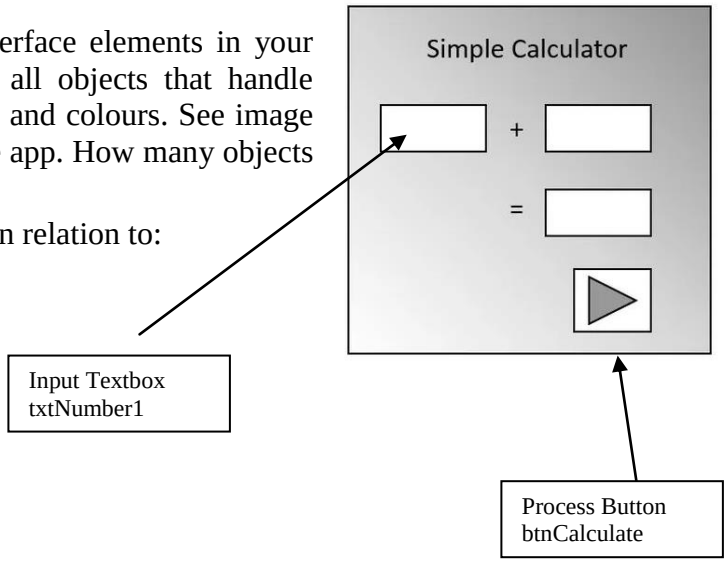
- Mock-Up Diagrams of Interfaces
- Data Dictionaries
- Object Description Tables
- Algorithms in Pseudocode
- IPO Charts

Mock-Up / Layout Design

It is important to display all the graphical user interface elements in your diagram. label ALL objects with object type and all objects that handle variables and data with their names, sizes, positions and colours. See image below for an incomplete Mock up of a mobile phone app. How many objects have not been labeled?

Also you can include justifications for your designs in relation to:

- Useability
- Maintainability
- Readability
- Validation/Feedback.



Data Dictionary

This design tool is crucial to your planning. Herein we define all the variables we will use in the solution, what names we give them, the data type each variable will handle, the size of the data in bytes, whether it is a global or local variable and a brief description.

Variable names usually need to follow suitable naming conventions, such as Hungarian Notation and/or Camel Case to make the code as easy to read as possible. For example, if the user to enter their Name into a textbox, it would be held in a variable called strName. Str indicates the data type string.

It is important to note the data types. If these have been incorrectly defined, you will experience errors in your executable code. For example, if you have been calculating the average of some values, you will not be able to set the variable to integer. If in the calculation of the average the returning value is not an integer the results will be incorrect.

Likewise, if you have an array with many types of data they will need to all be declared as strings. Once a value is accessed from the array, the data type can be changed within the code.

Data Dictionary Example

Variable Name	Data Type	Size	Scope (Global/Local)	Description
IntNumber1	Integer	3	Local	This is the first of two values entered
IntNumber2	Integer	3	Local	This is the second of two values entered
IntAnswer	Integer	4	Global	This is the sum of the two values entered to used elsewhere in the program.

Object Description Table

From your completed design layout, you can construct a more detailed table of objects. This identifies the object's name, type, method/event and description.

Using the Hungarian notation and Camel Case, a text box that reads in the user's name should be called "txtName". This kind of object allows the user to enter text into its "text" property. This means the object's purpose is "Method". A method is when an object responds to an event. However, when an object's purpose is to react to an event, such a button when the user clicks it, this is called an event object.

Object Description Table

Object Name	Object Type	Method/Event	Properties	Object Description
txtNumber1	textbox	Method	Set Text:Null	Reads in IntNumber1
txtNumber2	textbox	Method	Set Text:Null	Reads in IntNumber2
lblAnswer	label	Method	Set Text:Null	Displays IntAnswer
btnCalculator	button	Event: Click	Text: "Calculate"	When this button is clicked it will execute the calculation.

Algorithms

Algorithims are solutions to logic problems. In software development we use pseudocode to write algorithms. Pseudocode is like computer coding but easier to read and is often completed with English words for clarity.

All pseudocode algorithms must begin and end with "START" or "BEGIN" and "STOP" or "END"

Algebraic elements can be used: +, −, ×, /, mod and logical elements such as "and", "or" and "not".

Comparison elements =, ≠, <, >, ≤, ≥, <>.

When allocating data to a variable the symbol ← is used.

Algorithms allow for the outline of control structures used to design the solution.

Decision Control Structures:

START
IF Condition is true THEN
Action A
ELSE
Action B
END
	START
	CASE Selection
	CASE 1 : Do Action A
	CASE 2 Do Action B
	END CASE
	END

Iteration Control Structures

Counted Loops	(Pre-conditional Loop)	(Post- Conditional Loop)
START FOR counter ← 0 to 9 Actions NEXT END	START Do While Condition is true Actions End While END	START Repeat Actions Until a Condition is True END

Factors that influence design include:

- Usability – the measure of how well the solution is designed for the people who will use it.
- Affordance – the use of interface elements that are familiar enough to be used without instructions
- Security – sensitive data needs to be stored so that it is not compromised
- Interoperability – the system works easily with other systems already in use
- Marketability – will the design be popular

Measuring Efficiency of design:

- Time – how much time does it take to get the information required from the system.
- Effort– how quickly is it to learn how to use and how much effort is required to do what you need to do.
- Cost of file manipulation – how much effort is required by the user to complete the tasks.

Measuring Effectiveness:

- Completeness – are all the elements included so that the user does not have to go to a different system to get all the information they need.
- Attractiveness – people are more likely to use an attractive interface.
- Clarity interfaces need to be clear and readable, especially when on a small mobile phone screen.
- Accuracy – are the outcomes of the data processing correct?
- Accessibility – are there measures in place to assist people with limited eye-sight and other limitations.
- Timeliness – does the output become available when required, and does the app use the most recent data available.
- Communication of Message – does the interface clearly explain what the data is representing to the user.
- Relevance – does the system include any unnecessary features.

- Usability – the measure of how easy it is to use.

Interface Design Elements:

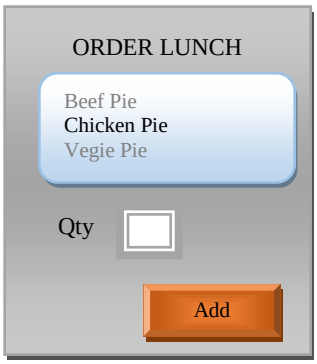
- Clear and Familiar design elements
- Responsive to the user
- Robust when user enters incorrect data
- Consistent in design allow for easy navigation
- Efficient – limits the amount of effort the user needs to make
- Scalable – able to be used on different screen sizes

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. Create an object description table from the Mock Up provided.



2. Using pseudocode create an algorithm that will test the user input for quantity.

TSSM
Creating VCE Success

Solutions to 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.

Object Name	Object Type	Event/Method	Description
cbxItem	ComboBox	Event	Selects the product to be added to the order and triggers code to allocate the price.
txtQty	Textbox	Method	Reads in the quantity of the item selected
btnAdd	Button	Event	Triggers the code to add the number items to the total required.

2.

START

```
Read in Quantity
IF (Quanity.datatype <> integer) THEN
        Message("Please enter an Integer")
END IF
```