

# Software Development Unit 3 Outcome 2 2011

Name:………………………………………………

**SD U3O2: Jackson’s Pond Emporium**

**Background**

*Jackson’s Pond Emporium* is a pond and fish supplier for pet shops and aquariums in South East Melbourne.

Recently you analysed some of the processes within Jackson’s Pond Emporium and created an SRS with the view to making some improvements – and there have been some changes since you were last on site.

**The Proposed Software Solution**

After analysing the practices that were in place to fill orders placed from pond supply stores and other clients, a SRS was written containing the following requirements:

**General description**

The required software solution will be used to manage the stock of fish, assorted aquatic creatures, plants and accessories at *Jackson’s Pond Emporium*. Shaun (in the warehouse) will primarily be managing the stock of fish (and other items) – although others can also do this.

The process for determining whether or not stock should be ordered is to be automated. The software solution will be designed to read a text file stored on the S: drive called ‘Stock.txt’. Inside the text file, will be the names of all the items of stock, the categories in which they are placed, how many of the items are in stock, their price and location in the warehouse (which is a single letter (A, B, C or D followed by a single number 1-9) and a reorder level. The reorder level is the quantity below which a restocking order should be placed. EG. If I have 20 in stock and the re-order level is 30, then I would need to order 10.

An example of this data is shown below (this is only a suggestion). You may need to add or alter the data to fully test your program.

Kohaku, Fish, 0, 25.92, B3, 3

Golden Poison Frog, Frog, 27, 3.95, D7, 20

Water Lillies, Plant, 25, 0.50, A4, 30

Sunken Pirate Ship, Accessory, 3, 10.99, B1, 2

You can determine the format of the text file and the way in which the program interacts with it – however, the text file needs to be read in when the program is started up. The results of the automatic re-ordering requirements should be shown on screen and an option given to save the output to file. A text file called ‘Reorder.txt’ should be produced that contains relevant details of those stock items that need to be re-ordered and the amounts required. It should also include the value of each stock item to be re-ordered ( Qty x price). The re-order file should also show the total value of the order and have some indication of when it was produced, either in the file or in the filename. Any orders with a total value over $500 need to be approved by manager, so some way of flagging this in the file should be included.

**Functional (F) and non-functional (NF) requirements**

* Read a text file called ‘Stock.txt’ (F)
* Produce and save a text file called ‘Reorder.txt’ on program close (F)
* Re-order file should contain the date it was produced, total value of order and notification if the order is over $500. (F)
* Display items of stock (F)
* Display items that need to be reordered and the quantity to be ordered (F)
* Clear and easy to use interface (NF)
* Reorder file should be able to be generated easily (NF)

**Constraints**

The software solution will be running on a mobile computing device yet to be determined.

There are a number of candidates – and all are touch enabled smart phone devices. For this reason, the interface should be confined to a screen size of 480 x 600 (portrait) and should be designed with touch interaction in mind.

* All input will be via touch (as this is a prototype, single mouse clicks will simulate touches of the display).
* Storage space on the device will be limited, so the software solution should take this into account.

**Scope**

* Will need to cater for all of the stock items at *Jackson’s Pond Emporium*. At present, the number of stock items is just under 200, so it would not be unreasonable to allow for more than this (but certainly not of the order of 400 or more).

**Some additional notes specific to the software solution**

* The data that you place in the text file can be considered to already be validated. Validation is not a part of this task.
* For testing purposes, it is only necessary to include a sampling of the stock at *Jackson’s Pond* *Emporium*, although you should be careful that the data you do include covers the situations /combinations necessary to fully test the functioning of your program.

**The Task**

Represent a software design and apply a range of functions and techniques using a programming language to develop a prototype solution to meet a specific need. **Tasks 1 & 2 must be submitted for marking by the end of the first double period.**

**Complete the following:**

1. Create a data dictionary to describe the format of the text files that will read in and produced by the software solution.
2. Construct an algorithm to represent the operation of the software solution.
3. Identify evaluation criteria appropriate to the solution requirements.
4. Construct a prototype software solution that meets the specifications in the SRS, ensuring that the prototype solution:

* contains efficient and effective programming techniques.
* contains efficient and effective use of data types and data structures.
* has intuitive and consistent navigation.
* contains clear and appropriate internal documentation.
* has a logical and attractive user interface.
* produces the correct output.

1. Create a testing table that contains a number of tests, the expected outcome of each test and the actual results of the tests. The tests should cover both navigation and the solution itself.

**Scope of Task**

**Dates: 24/5- 1/6**

**Time allowed: 8 periods.**

**Resources:** You may refer to class notes, programs, your text book and practice SACs. You may **NOT** use the internet or email during the SAC, nor seek or give assistance to any other student.

**Authentication:** No SAC work is to be undertaken out of class time. The teacher must be able to authenticate that all work submitted belongs to that student.

**Saving work:** All SAC work is to be saved on the S: drive coursework SD folder. No work is to be saved on your USB or H: drive. Make sure you save regularly throughout the SAC. You should set up the saving options in VB to point to S: drive.

**Marks:** This SAC is worth 60% of Unit 3 marks.