APPLICATION DESIGN PROJECT PART B

SUBMISSION 1

DAN
GRIZLI777

Contents

Introduction

In this document I have outlined the scope for the WoodStocks project, as well as discussed how WoodStocks will be able to implement usage of the new software as seamlessly as possible. I have also explained why and how the software is going to be of benefit to WoodStocks.

I have included UML diagrams to give an overview of how the program will function and be constructed.

I have also presented a risk analysis matrix and suggested alternatives within the current project constraints, along with a list of assumptions and dependencies.

Project scope and business efficiencies

SkillAgeIT will be constructing and deploying an application for WoodStocks. The application will allow the user to amend and input stock information into an existing file, without alterations to the format of the existing file.

By implementing the software from SkillAgeIT, WoodStocks will be able to prevent the double handling of information, by no longer having to print stock lists to amend the data and then re-enter the information. Because the stock data will now only need to be input once, without printing, the program will save time and also reduce the potential for erroneous data entry.

Technical and operational feasibility, and how the project will fit within the organisation

The project to be undertaken by SkillAgeIT is relatively minimal, as the software is only required to perform basic tasks. As such, the financial cost put forward to WoodStocks will be relative to the requirements of the project.

WoodStocks and its' employees have, and are already familiar with the basic use of computers. SkillAgeIT will be designing the software with simplicity in mind, and it will require minimal training for employees at WoodStocks to utilise the new software.

There is nothing about the software which will interrupt the workflow of WoodStocks in any detrimental way. The stock information is already being input into a computer program, the only foreseeable difference is that the information only needs to be handled once which will speed up workflow and reduce the potential for error.

Functional and related non-functional processes

Functional processes would best be described as the necessary abilities that the software is required to have. If the software does not meet the functional process or requirements, then it would be deemed unsuitable.

Non functional processes are the abilities that the software ought to have. They are things that will improve the usability and experience for the user, but that are not strictly necessary for the software to perform its' duties.

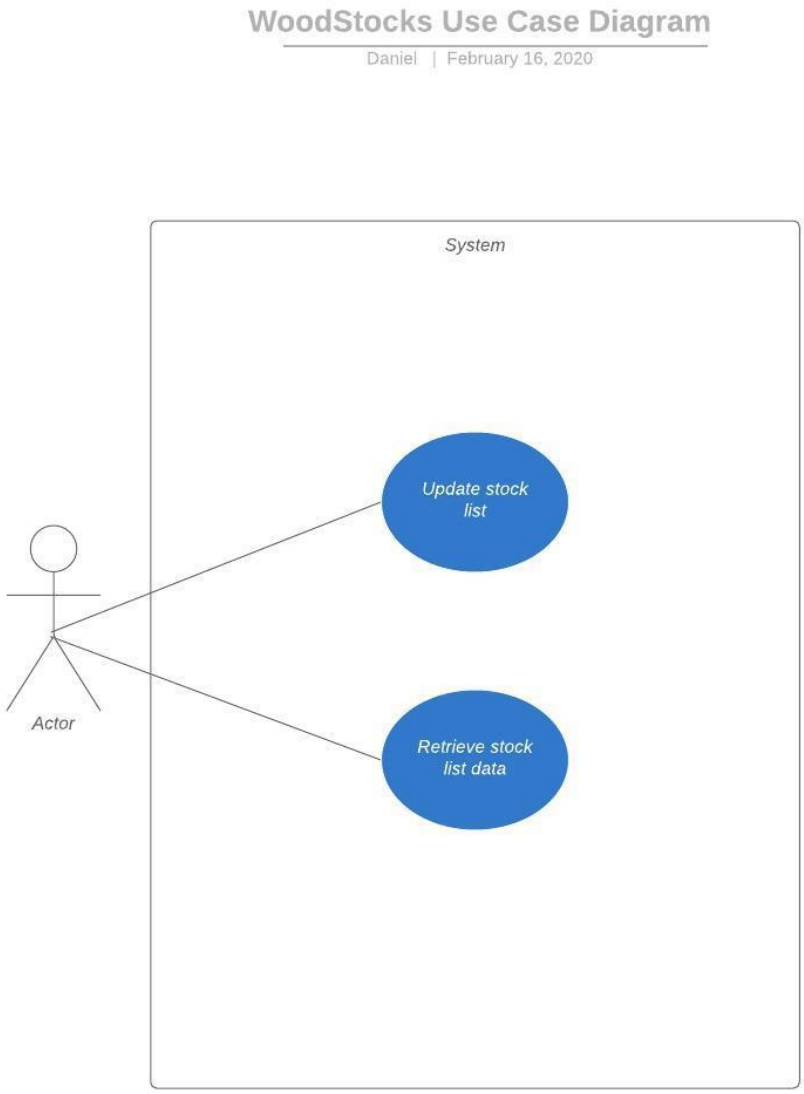
Functional Requirements:

- Display the contents of a text file
- Facilitate alterations to the data fields
- Keep text file in the same format
- Read and write to C:\StockFile\
- Be able to arrange item order according to item code, current count, on order

Non – functional requirements:

- Simple, graphical user interface
- Allow 24/7 access

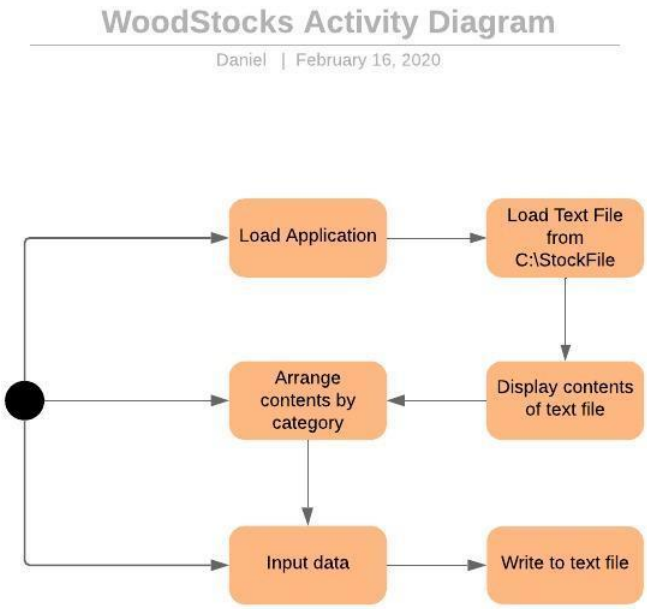
UML use case diagram



This use case diagram displays the main intended scenarios for use of the program.

The actor in this diagram represents a human user of the program, most likely an employee of WoodStocks. Everything within the rectangle takes place within the program. The line connecting the actor to each use case represents the relationship between who it is that is using the program, and what it is that the program is being used for.

UML activity diagram

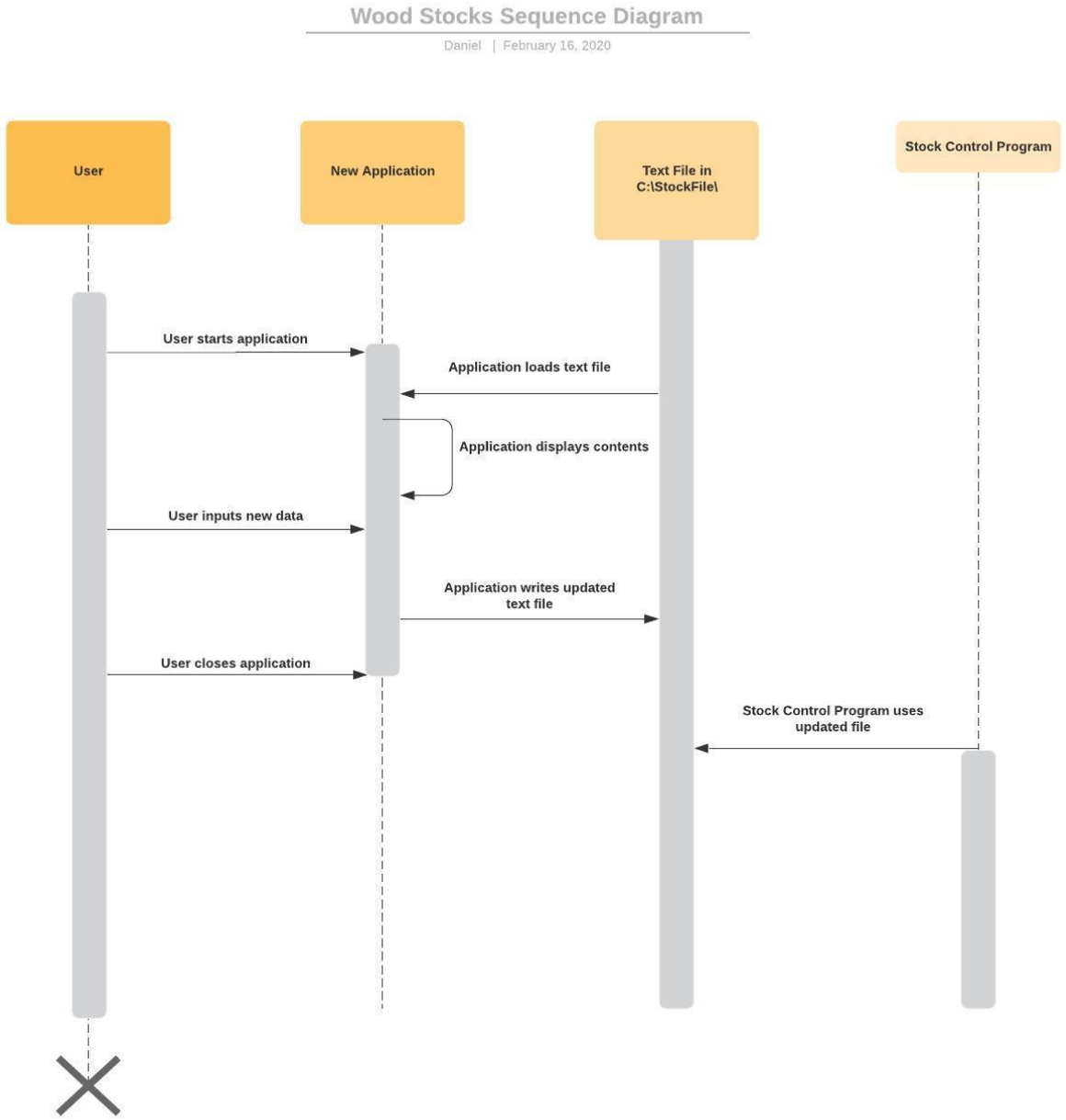


This activity diagram displays the order of operations for the program. It can be perceived of as a flow chart. The black circle represents the user, and the system functions are presented within the orange rectangles.

Because the user is connected by a line only to “Load Application”, “Arrange contents by category”, and “Input data”, this accurately represents the relationship that a user will have with the program. For example it is not the user who will load the text file into the program; it is the program itself that will load the text file.

This diagram shows the functional relationships of the different aspects of the program.

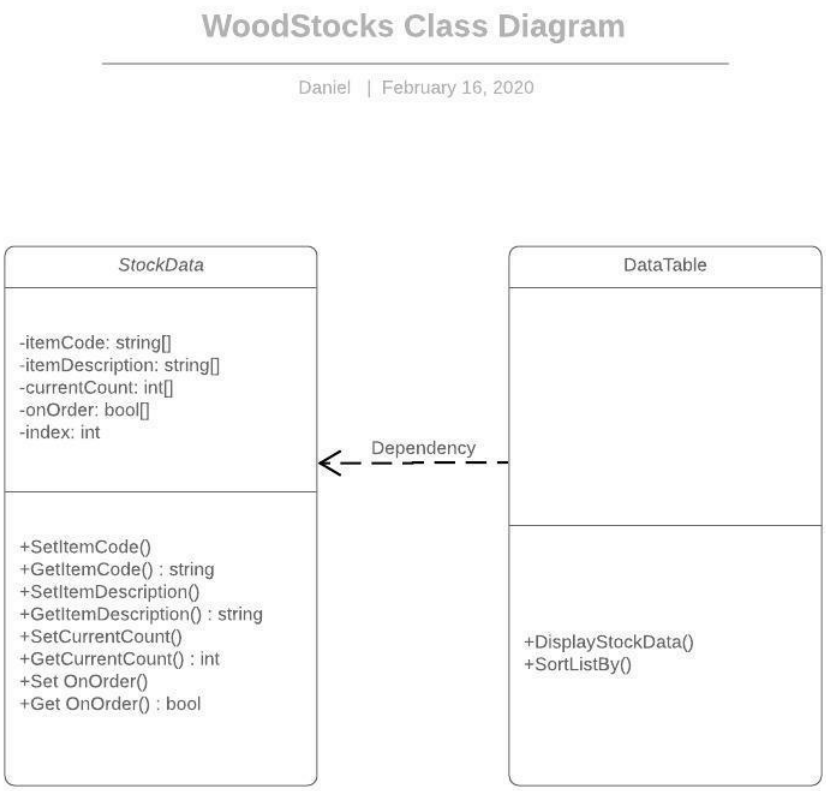
UML sequence diagram



This Sequence Diagram displays the sequence of how operations are carried out. The process starts at the top, and progresses towards the bottom.

It displays the order of relationships and interactions of each part of the overall system, as well as the responsibilities.

UML class diagram



Behaviour and state of classes

An instance of the StockData class will act as a data field and will be populated by the text file.

The DataTable class will contain the methods needed to populate and display the contents of StockData onto the screen. It will also contain the methods needed for the user to be able to sort the item list in different ways.

Collaboration between classes

The DataTable class is dependent on the StockData class, because without an instance of StockData with values in each field, the DataTable has no information to populate the display table.

Generalisations and specialisations within classes

Generalisation in this context refers to a way of organising classes. A super class (parent class) will contain the generalized properties and methods of a type of object. A specialised class (often referred to as a sub-class) will contain methods and data fields from the parent class, but will also instantiate further detail to refine and specify further data and methods.

Applying the principles of aggregation and composition to refine the class diagram

Aggregation describes a relationship where a certain class would contain instances of another class or classes.

Composition describes a relationship where a certain class would be made up of a composition of other classes, and without those classes it would no longer exist.

Alternative solutions against project constraints, Assumptions and dependencies

If for any reason, the software cannot be developed using the C# programming language, it may be completed using either the VB.NET or C++ languages and will still function within the .NET framework.

Assumptions:

WoodStocks computer are running .NET Framework
WoodStocks employees are capable of performing basic computer functions
WoodStocks budget is sufficient for the scope of the project

Dependencies:

Successful review of prototype (Finish to Start dependency)

Project risk analysis and future requirements

			Impact			
			0 Acceptable	1 Tolerable	2 Unacceptable	3 Intolerable
			Little or No Effect	Effects are Felt but Not Critical	Serious Impact to Course of Action and Outcome	Could Result in Disasters
Likelihood	Improbable	Risk Unlikely to Occur				Software failure resulting in loss of company information
	Possible	Risk Will Likely Occur	Small amount of computer downtime while the software is being deployed	Employees struggle to learn how to use the software		
	Probable	Risk Will Occur				

To mitigate the risk of a software failure resulting in the loss of stock information, the file containing the information should be backed up.

To mitigate the risk of computer downtime, the software could be deployed during non-business hours, although it seems unnecessary to do so because the implementation ought to be simple.

To mitigate the risk of employees struggling to use the new software, we produce well written documentation which will instruct them on the use of the software.

Sign off for client approval

Project Client Acceptance and Sign-Off Form

Project Name:	WoodStocks Stock Counter
This Document is Issued by:	Daniel Stride
Date:	17/02/2020

Dear WoodStocks Inc.
This form is to confirm that you give your approval for the commencement of development of your Stock Counting software, to be built and deployed by us, SkillAgeIT.

Software Requirements:
The product analyst report collected by Gary Pearson specified the following requirements: The software is required to read a text file containing stock information, and allow the user to amend the data fields and re-arrange the data table to be able to sort the list by the different categories. It is also a requirement that the software is able to write to the existing file, or write a new file and keep the same format consistent through each update to the file residing in C:\StockFile\

Name:

Position held in company:

Signature:

Date:

References

https://www.inloox.com/company/blog/articles/a-guide-to-dependencies-constraints-and-assumptions-part-3-project-assumptions/

https://www.inloox.com/project-management-glossary/dependencies/