Feedback for Software Engineering Fundamentals

Based on some of the common themes from previous submissions and questions here is some further guidance for the assignment.

**General**

* Include Assignment Frontsheet with details completed.
* Ensure the task numbers are displayed in the title e.g. Task 1 – High Level Design
* Provide short descriptions at the start of each section to provide a clear narrative for the reader
* Label all diagrams e.g. Figure 1 – Use Case Diagram for “My System”, below the diagram and centre-aligned
* Include all key information in the main body of the document, Appendices are only for supplementary (extra) information e.g. additional company context.
* Ensure diagrams are big enough to be readable
* Be careful when referring to Data Type as they are different in SQLite than Python!

**First and Third Person**

All academic work (and business reports) should be written in the third person, see examples below. The only exceptions to this are ‘reflective’ tasks where you are writing from a personal perspective and reflecting on the experience ‘as you see it’ – Task 4 in this assignment is an example of a reflective task.

* First Person Perspective:  I, we, me, us
  + **I** will explain how the CRM system has supported the business strategy
  + **I** have found the benefits of CRM, expressed by Yeboah (2019), outweigh the disadvantages.
* Third Person Perspective : he, she, it, they, him, her, them
  + An explanation of CRM system will show how **it** has supported the business strategy
  + **It** could be argued that benefits of CRM, expressed by Yeboah (2019), outweigh the disadvantages.

**Task 1**

At the start of this task please provide a brief overview of what the application is for and the business context that it will operate in e.g.

*“This design outlines the intended operation of a basic Asset Management system. It will store basic details of assets and provide visibility of all the IT assets within the business. The scope is currently based on any asset that has an asset tag with barcode currently attached. The system Use Cases are shown in Fig. 1, below.”*

Please note Fig. 1 here is just a generic example.

An UML - Use Case diagram looks like this: Diagram, schematic

Description automatically generated

Figure 1 – Use Case Diagram

This isn’t meant to show program flow but all the ‘things’ the user will be able to ‘do’. We expect each Use Case to have a verb-noun expression i.e. DO-SOMETHING nice and simple but shows the stakeholders all the things the app will be able to do and all the actors (including file systems or DB systems) e.g. “Select Option”.

An UML - Activity Diagram (think of this as a flow diagram) will show the simple flow of the application, for example:

Chart

Description automatically generated with medium confidence

Figure 2 – Activity Diagram

An UML – State Machine for an Asset Management System will show the various ‘statuses’ the asset record can move through. These ‘business rules’ can then be implemented/enforced by the code.

Diagram

Description automatically generated

Figure 3 – State Machine Diagram

A very basic navigation diagram looks like this:Diagram

Description automatically generated

Figure 4 – Display Screen Flow

# Data Store Specification

A basic Data Stores specification would consist of a table along the lines shown below. The Data Store itself is either a .csv text file or the SQLite DB.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field Name | Description | Data Store Type | Python Type | Validation Details |
| Your field name 1 |  |  |  |  |
| Your field name 2 |  |  |  |  |
| Your field name 3 |  |  |  |  |
| Your field name 4 |  |  |  |  |
| Your field name 5 |  |  |  |  |

Field Name – this is the variable name in the Python programme e.g. asset\_id, asset\_tag, description, asset\_type, cost, date\_purchased

Description – text to describe what the field is – “Contain the unique value of the asset tag attached to the asset”

Data Store Type – this is how the data is being stored - if using a csv text file this will be always be text – if using SQLite DB this will be either text, integer, real or blob

Python Type – this is the data type in Python e.g. string, integer, float, boolean, datetime

Validation Details – this is where you specify the rules to determine if a field is valid or not e.g. date\_purchased must be dd/mm/yy, asset\_tag in format xxx-xxx-xxx, asset\_type must be “Leased’ or “Purchased”

# Test Plan

Test plans can come it lots of different styles but each test will need to have at least:

* Test ID
* Test Purpose
* Test Steps
* Test Data (could be included as part of Test Steps)
* Test Outcome
* Pass/Fail
* Test Evidence (best done as a link to screenshot in the Appendix)
* Required Actions