

College of Science and Technology

School of Science and Technology

# SOFT30121: Advanced Analysis and Design

# Systems Analysis Design and Implementation

*By*

N0683858: Hamid Mujtaba

T0084963: Haowei Goh

N0688119: Elliot Harding

N0756079: Steffan Walker

N0704050: Michael Abraham

N0772082: Hamed

NTU Stores Management System

SOFT30121: Advanced Analysis and Design i

Systems Analysis Design and Implementation i

1. Introduction 3

1.1. Purpose 3

1.2. User Characteristics 4

1.3. Assumptions 5

1.4. Scope and Constraints 5

1.5. Glossary of terms 7

1.6. Overview 7

2. Functional Requirements 7

3. Non-Functional Requirements 10

3.1. Usability requirements 11

3.2. Reliability requirements 11

3.3. Performance requirements 11

4. Interfaces 14

5. Use Case Modelling 30

5.1. Use Cases 30

5.2. Misuse cases 30

6. Project Plan: 31

References 33

7. 33

Bibliography 34

8. 34

# Introduction

## Purpose

Specify the purpose of this Software Requirements Specification. The SRS fully describes the external behaviour of the application or subsystem identified. It also describes non-functional requirements, design constraints, and other factors necessary to provide a complete and comprehensive description of the requirements for the software.

The purpose of this Software Requirements Specification document is to propose a system which will help the NTU staff to manage the stores. In an interview with Lisa, it was made clear that they currently do not possess a system which allows them to monitor the quantity of the products being consumed or to allow the (consumer) staffs to efficiently take the products. Instead, they write down the details on a piece of paper, from which the data is manually stored in 2 separate systems by the management staff; QuickBooks, which is responsible for creating an invoice for each department, and Sage, to update the database. The 2 systems mentioned are difficult to use, in addition to being time consuming and lack a lot of features making it hard for the staff to manage the stores. The proposed system will overcome this by combining Sage and QuickBooks along with providing additional features, such as the functionality to scan the products using external hardware, which will make it easier for the consumer staff to take items and the management staff to manage, as the system will update the database automatically. According to Lisa, they also find it difficult to keep track of stockout as the current system(s) do not update the database in real-time; they are made aware of the stock quantity only after creating an invoice. The current user interface is very complicated to use which can be a problem especially when training new employees. The proposed system will overcome these issues by implementing a database which will be automatically updated. It will include a notification feature to notify the staff when they are low on stock. The team aims to develop a system with a very simple user interface which doesn’t require much input, thus anyone will be able to use it. She also said they use their own alpha codes to identify products, which is hand written. Ultimately, Lisa wanted some way for the consumer staff to just scan the products instead of writing their details on a piece of paper which can be an issue when there are more than a few consumer staff in the store at the same time. The proposed system will have a feature which will allow the admin to create custom alpha codes for the products when registering them into the catalogue; the system will not have the feature to allow the user to scan products directly, however, it will include the functionality to read barcodes with the help of external hardware.

The system is required to be able to; manage sock, create invoice for each department, create log of products taken by (which) staff, add new products on the catalogue, register products received from shipments, provide directions to help staff find items, notify the staff about important events, such as low stock, or shipment date. The system will deploy a very simple, easy to use UI with minimal input which will require no prior knowledge on using management systems. It will also provide staff with a ‘basket’ feature which will allow them to take bulk of items at once. There will also be an option for returning items, so if staff accidently took more than required, they can simply decrease the quantity on the checkout interface. The catalogue will have pictures with a brief description of the items to help consumer staff identify them easily. All the data will be backed up on an external (cloud-based) database to avoid data loss. The system will also have restriction on some functionality depending on the user role; it will allow the admin to update roles.

## User Characteristics

Create general customer profiles for each type of user who will be using the product. Profiles should include:

· Student/faculty/staff/other

· Experience

· Technical expertise

· Other general characteristics that may influence the product

The proposed product management system will include three kinds of users. The system privileges/features available to each of these will vary, as will their experience and knowledge of product management. The three types of users will include:

**Consumer**

The consumers will be comprised of university researchers/lectures who require the products available at the store. These users will be limited to removing and returning products to the store. They will not require much experience with virtual systems due to their limited usage of the system and the intuitive user interface which will walk them through the product removal process. The consumers will only need to understand the basic fundamentals of computer usage and graphical user interfaces in order to utilise the system. Interface basics such as clicking a plus to increment the quantity of a product, back arrow to move to the previous page etc. will be all that is needed of this user.

**Staff**

The staff will be the employees working in the store that are in charge of managing stock, receiving shipments and assisting consumers. This type of user will have greater system privileges than the consumer as they will be responsible for the systems store management features. The proposed system is designed to automate a large amount of the manual work needed in the current system and as such the work load is lessened. However, the staff will still be required to input data into the system such as shipment codes so that the system can update the log and the stock database. This will require staff to have some experience with data input in addition to graphical user interfaces. Experience with systems such as MS Excel will transfer easily to the proposed system.

**Admin**

The admin will be the store employees that have greater control over how the store is managed and run. The admin users will be responsible for managing the catalogue of items available to the consumers in addition to viewing logs of removed products and received shipments. Because admins will have greater control over the store as a whole, they will require management and stock control experience in order to fully utilise the system. The level of technical expertise will be similar to that of the staff.

## Assumptions

List any assumptions that affect the requirements, for example, equipment availability, user expertise, etc. For example, a specific operating system is assumed to be available; if the operating system is not available, the Requirements Specification would then have to change accordingly.

## Scope and Constraints

A brief description of the software application that the Software Requirements Specification applies to.

Also describe any items that will constrain the design options, including

· Parallel operation with an old system

· Audit functions (audit trail, log files, etc.)

· Access, management and security

· Criticality of the application

· System resource constraints (e.g., limits on disk space or other hardware limitations)

Other design constraints (e.g., design or other standards, such as programming language or framework)

The system is intended to be comprised of two main components. Firstly, the software, designed as an executable, in which all the functionality and user interactions of the system will be housed. And Secondly, cloud storage-based database which will store all the systems corresponding data.

Design of the executable will be done using C# with windows forms using the IDE Visual Studio. Using windows forms limits the amount of restrictions in terms of design due to its high flexibility in this area.

The database will utilize SQL (Structured Query Language) in order to store and retrieve its data. As for cloud storage, the hosting website ‘Gear Host’ will be used, thanks to its free trail size databases. In order to link the two components together, php hosted on the gear host server with the database will be called by functionality of the executable.

While multiple instances of the system will be able to run at once, the stores will have a limited number of computers in which the software can run. This constrains the number of users who can use the system simultaneously. This however is not so much of an issue since users can wait for their turn to use the system.

Given that the data for the system will be stored within the cloud, the disk size of the computer on which the system will run is also a minimal constraint. Generally, software applications of similar complexity take up much less than a single gigabyte of disk data.

The size of available cloud storage will be constrained by cost. An initial test database (with minimal capacity) can be setup for free, but afterwards payments may be required in order to expand its capacity, unless of course NTU servers can be utilized for storage.

Given that the storage format of the previous systems product and user data is unknown, transfer to the new system may prove difficult. As a last resort the inventory of the store may have to be entered in manually into the new system, a time-consuming process.

Since data for the system is stored online in the cloud, for the system to function correctly, an internet connection would be required. As well as this, the transfer rate of the internet connection would be an additional constraint for the system.

The application will be written using windows forms and the .Net framework and therefore it will be compiled in an .exe format. This means that the operating system in which the software can run will be constrained to windows only (Or Linux with Wine, a windows emulator).

Due to different priorities in terms of usability for different users of the system, the UI will be constrained in terms of its design. Design layouts will need to take this into account. On top of this certain UI elements and features will need to be hidden for lower priority users.

## Glossary of terms

Outline any acronyms, abbreviations and definitions that will be used throughout the document

## Overview

What does the rest of this document contain/how is it organised?

This rest of the document contains details about the proposed management system requirement/non-functional requirements which are then further split into 3 categories; the requirements the system must have, should have and could have, which were written after carefully considering user requirements and needs. The non-functional requirements are split into, usability requirements, reliability requirements and performance requirements. The following section has use cases and misuse cases for the management system where the main functionality the system will perform and the functionality which the system will not perform is discussed. The section will also include use case diagrams using the UML notation for a visual representation of how the system and the actors will interact. The last section is about the project planning, where the details on the agile methodology the team used is discussed and the process of how we plan to implement the system. This section also includes the strengths and weaknesses of each individual member in the group.

|  |  |
| --- | --- |
| **Intro** |  |
| The problem the system is to solve is explained is understood |  |
| The purpose of the document is outlined clearly, and consistent with the rest of the document |  |
| The users of the system have been described and discussed |  |
| All major assumptions made in the rest of the document have been stated |  |
| The scope is well defined, and realistic for the project. Relevant hardware and software constraints identified and described |  |
| The glossary serves as a useful aid in reading the rest of the document |  |
| Significant consideration of each aspect of the problem is evidence, and discussed |  |
| The Users’ technical expertise has been discussed |  |
| All assumptions have been stated, and brief details on how the document would have to change if these were to change has be |  |
| Scope is bulletproof, there is no ambiguity in what is in scope. Constraints discussed in good detail. |  |
| As for First plus:  Evidence of a user interview |  |

# Functional Requirements

These should be described in enough detail for the designers to design a system satisfying the requirements. These should also be testable.

They should also be organised appropriately (could be by feature) and prioritized (could be by MoSCoW).

A good requirement is:

* Correct
* Unambiguous
* Complete
* Consistent
* Ranked for importance
* Verifiable
* Modifiable
* Doesn’t specify any particular design
* Traceable

## Usability requirements

Must

**2.1.1: Store the inventory of the store’s stock**

The system must utilise a database to store information on the products being sold, their quantities and the alpha codes used in store to uniquely identify products. This database should update in real time when products are removed from or returned to the store. Keeping a digital count of stock is more accurate and robust than written logs.

**2.1.2: Display currently available stock**

The system must display, in a catalogue style, the stock currently available in store. The information should include the products name, the products alpha code, a picture of the product and the quantity available. Since the consumer can find whether the product they require is stocked immediately, there is less time wasted searching and asking staff members if the product is available.

**2.1.3: Use a virtual basket when buying products**

When consumers go to remove an item, it should be added to a virtual basket along with any other products they might want to remove. This basket can then be “Checked out” when the consumer leaves the store and the products removed be registered in the system. Before checking out, the consumer should be able to add and remove products they have in their basket. Virtual baskets are used by a majority of online retail sites which makes the system more relatable to consumers and is also a quick and clear indicator of what a consumer intends to take from the store.

**2.1.4: Allow products to be returned**

The system must allow products to be returned that have previously been removed from the store. When a product is returned it will be inspected by a staff member before the return is registered which in turn updates the stock, the log and the invoice. Consumers may find they no longer need the product that they removed from store or that it was inspected incorrectly by staff. For this reason, a feature to return any product that was removed from the store must be implemented.

**2.1.5: Allow catalogue of products to be changed**

The admin users must be allowed to add new products and remove old ones from the stores catalogue of available products. Since the needs of consumers inevitably changes as time goes on, the ability to add new products to sell and remove old ones will be implemented to keep the store adapting to their customer’s needs.

**2.1.6 Create invoices**

The system must create an invoice based on information of what products have been removed, the price of the products and the department which removed the product. The invoice should be created at the end of the month. By generating invoices automatically, there is less workload on staff and a decreased chance of invoices being completed erroneously.

**2.1.7: Allow users to login to the system**

The system must facilitate logging in for three types of users. These users include:

* **Admin:**

All the privileges available to staff in addition to control over the catalogue

* **Staff:**

With ability to receive shipments and inspect returned items/shipments

* **Consumer:**

Only allowed to remove and return products from the store

Should

**2.1.8: Convert supplier units to consumable units**

The system should convert the unit a product is bought and received in, to the unit that the product is consumed in at the store.

E.g. 1 Box of pens should be converted into the number of individual pens a box contains as pens will be removed from store individually.

Performing this conversion through the system reduces erroneous input and time spent processing deliveries.

**2.1.9: Register received shipments**

The system should allow staff members to input shipment data, such as supplier codes and quantity, so that the store’s inventory updates with the stock contained in the shipment. Allowing the stock to automatically update with the contents of shipments reduces the workload of staff and allows shelves to be refilled quicker.

**2.1.10: Allow stock inspection**

Stock that has arrived from shipments or been returned by a consumer must be stored in an inspection database separate to general stock. When stock has been inspected by a staff member it is then moved to the general stock database. Since products can be damaged upon delivery to the store or by consumers trying to make a return, it is important to have some method of inspecting products before they are re-introduced to the stores stock.

**2.1.11: Display tags for important information**

The system should display, on the product page, any important information relating to the product in question. Products which are toxic, explosive, radioactive or pose any kind of health risk should have this information displayed clearly and readily visible at the top of the product page. While products that pose health risks are usually labelled accordingly, it makes sense to include this information on the product page before the consumer attempts to handle a potentially dangerous item.

**2.1.12: Track items that expire**

Products that can only remain on store shelves for so long should be tracked through the system so that when expiry dates are close, the staff members are notified. Implementing this feature is important for product rotation whereby products with soonest expiry dates are placed closer to the front of a shelf and vice-versa.

Could

**2.1.13: Provide directions to products**

The system could display a map of the store with a marker indicating the location of the desired product within the store. This feature will allow store management to adjust product placement within the store and still provide a means for consumers to find their desired product.

**2.1.14: Remind staff about deliveries**

The system could remind staff members with notifications about upcoming shipments/deliveries. The notifications would include the delivery’s time, products being delivered and any additional information such as whether special handling equipment is needed. Notifying staff of upcoming deliveries allows time to prepare for substances like gases that could require special handling procedures.

**2.1.15: Scan barcodes of products**

The system could allow users to scan barcodes of products to remove the need for manual input of data into the system. Barcodes would be scanned when consumers add products to the basket and when staff need to register received shipments as part of the store’s stock. Scanning barcodes would reduce the amount of data input work done by both staff and consumers and reduce accidental adding of products to the basket.

## Reliability requirements

Must

**2.2.1: Reliable database language**

The system should use SQL for the databases it uses. SQL is an industry standard language designed for building reliable and robust databases which makes it a solid choice for the system. Usage of a good language will help make a database that is expandable and resistant to incorrect data input.

**2.2.2: Log the removal and return of products**

The system must keep a log of what products have been removed/returned by storing the products name, product alpha code, date and time of occurrence, the individual involved and their department name. Keeping a log of the removed products will provide something to calculate invoices from.

Should

## 2.2.3: Backup data externally

## The system should backup any data it is storing locally, into an external storage location to prevent data loss. The data should be backed up to a cloud-based drive like OneDrive to ensure there is an existing backup in the event of data corruption or accidental deletion.

**2.2.4: Provide low stock warnings**

The system should notify staff members when the stock of a particular product is low so that more can be ordered in time. By promptly informing the staff of low stock, there is a lessened chance that the product a consumer requires will not be available which improves the stores reliability.

**2.2.5: Log received shipments**

When shipments are received and registered through the system, they should be logged in a database. The log should include the shipment’s Order Head with details about the shipment itself and the Order Line with details on the products and quantities included in the shipment. Logging the shipments received will give management a clear view of what products are being re-ordered frequently which is very useful information.

Could

**2.2.6: Offline Buffer**

The system could include a buffer by means of storing a local copy of the product catalogue so that products could still be taken out if the system was not connected to the internet. When the system regains an internet connection it would then update the stock level and product removal log. This would prove useful as consumers would still require products from the store if the universities internet was disrupted.

## Performance requirements

Must

**2.3.1: Memory efficient language**

The system should be written in C# due to the languages strong memory management capabilities. This will improve the systems performance and reduce the users waiting time for operations to happen.

**2.3.2: Real time stock update**

The systems should update the stock database as soon as a product is checked out of the store. Updating in real time provides the staff with information that is current and provides a more accurate representation of the stores current stock level.

Should

**2.3.3: Responsive UI**

The graphical user interface should be responsive and receptive to the action’s users make. Interactions such as clicking the increment/decrement quantity buttons should quickly update the number of products in the basket. By making the interface respond swiftly, there is a lessened chance that users will make an erroneous click in anticipation of the interface updating.

Could

**2.3.4: Multi-threading**

The system could make use of multi-threading to process multiple actions concurrently. Considering there will be multiple users making use of the system, a multi-threaded program would be able to handle user request quicker and more efficiently.

# Non-Functional Requirements

Describe the characteristics of the system. Explain the rationale of each of these to aid design choices later. Be as specific as possible. Not all these requirement types may apply, so only use ones appropriate the system.

From the description above the best approach seems to be a format for each requirement, with Explanation, Rational, and priority rating (from previous).

## Usability requirements

…

## Reliability requirements

…

## Performance requirements

…

* Store inventory of current stock

*This is currently implemented through Quickbooks. System should update stock in real time. (Current system only updates stock at then end of the month via invoice)*

* Create invoice

*This is currently implemented through Sage. The invoice should be created from data on what products have been taken, the quantity and the department they have been taken from.*

* Create log of products being taken

*Logging the products being taken, and who has taken them should allow to create an end of month invoice*

* Display currently available stock
* Unit Conversion (Boxes to Items)

*The system should convert the codes used by the supplier into an amount that is used by the system. E.g. a supplier code AGH43579 Could mean 1 box of 100 packs of individual pens.*

* Register products received from shipments

*The relevant staff members should be able to input the code used by the supplier which then updates the stores current inventory*

* Database to cross reference codes used by the store and the supplier

*The store uses alpha codes to identify specific products. A different code is used by the supplier to identify the products. The system should allow the admin/relevant staff member to input the suppliers code and have the inventory update with the relevant code used in store.*

* Provide directions through the store to a desired product

*Optional feature. Would be useful in particular when the store is busy. However is not a crucial part of the system.*

* System could remind relevant staff of when the next delivery is/details about what is being delivered.

*Optional feature. Not desired but would be useful for when customers require something out of stock and want to know when it will be back in*

* Return feature for products to be returned

*The feature should allow products to be returned which will then update the inventory and the corresponding invoice for the department that returned it.*

*Must be exclusive to staff members to avoid consumers returning damaged goods*

* VAT Option

*Must allow admin to select whether products being added to the catalogue have VAT or not*

* Allow admin to update the catalogue of items being sold

*The admin could have the ability to add/remove products that are being sold in the event new products are desired or old products need to be removed*

*(Ask Nigel)*

* Low quantity reminder

*The system should notify the relevant staff members that a products stock is low so that more can be ordered in*

* Log in

*Facilitate the login of an admin, staff member and regular consumer. Staff may update the inventory and receive shipments. Admin has more control over things such as the catalogue of items. Regular consumer may only take out and return items to the store. Apprentice’s cannot*

* Inspection Stock

*The system must store products in an inspection database first and can only be added as general stock after a staff member has inspected. Staff member can have ability to add products to the general stock after inspecting an item.*

* Bar Code Scanning

*To improve ease of use, a barcode scanner could be implemented that removed the need to input product data into the system manually.*

*Optional feature. Not necessary for system completion but would improve ease of use.*

* Product Tags/Categories

*The system should display important information for products that could be deemed dangerous such as toxic chemicals/explosive gases.*

* Basket System

*The consumer should be able to see their current items they plan to take out with pictures of the items, quantity, unit. They should be able to remove items and add new ones to their basket before they finalize their removal and “Checkout”. (System is very similar to the scan and go used at Tesco)*

*If barcode scanner is used this could update the basket by scanning multiple times for multiple items.*

* Track items that expire

*Certain items cannot remain on the shelf for certain periods of time and as such should be monitored and kept track of how long an item has left on the shelf.*

*(Ask Nigel)*

* Track Shipments Received

*Must include:*

*Order Head*

*Supplier Name*

*Supplier Site Name*

*Supplier Remit to address*

*Order Number*

*Order Date*

*Requested Date*

*Promised Date*

*Goods and Services Total*

*VAT*

*Invoice Total*

*Order Line*

*Supplier item number and/or universal product code*

*Item description*

*Quantity*

*Unit of Measure*

*Unit of Price*

*Extended Price*

*E5 will provide this information which the staff member must then input into the system to keep a log of the shipment.*

|  |  |
| --- | --- |
| **Requirements** |  |
| The requirements are detailed and organised in a way that is clear to read, and appropriate for the project. |  |
| Requirements don’t only concentrate on main stakeholder and cover all key requirements needed to meet the goals of the system. |  |
| The requirements should also be prioritised using MoSCoW or similar. |  |
| Full set of requirements covering the main stakeholders and their needs. |  |
| No functional aspect of the system is left ambiguous, and all the requirements fit the quality attribute described in upper second. |  |
| The requirements refer to the needs of the users outlined in the introduction. |  |
| Non-functional requirements addressing a range of system characteristics have been included and organised appropriately. |  |
| No requirement has been identified incorrectly |  |
| Requirements are present that directly address user characteristics. |  |

# Interfaces

Describe the logical characteristics of each interface between the software product and the users, and any external systems. This may include sample screen images (consider using balsamiq!), any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Sections User interfaces, Software interfaces, Hardware interfaces.

**3.1 User interfaces**

Design

The user interface is to be designed in a simplistic manner in order reduce the learning curve of the system. Thanks to controls in the UI following a similar layout to similar existing systems, staff should have little difficulties transferring their knowledge of previously used systems.

Controls such as buttons will have black outlines to provide clarity in their purpose and location as well as allow colour blind users to distinguish them from other controls. Any of these controls which are used on multiple pages will be situated in the same location for each page to be consistent, reducing cognitive load for the user.

Most pages of the system will have a distinguishable help button. Clicking on this button will open a popup describing the purpose of the page and of any ambiguous elements it may contain.

Colour Palette

Surface & background colours



Error, message & action colours



The chosen surface & background colours have been selected as weak toned colours as to not distract the user from more important UI elements. Two variants of the surface & background colours allow contrast between UI elements, such as distinguishing a background of a UI element and the system background, while keeping the same overall theme consistent.

In contrast, error, message and action colour tones have been chosen as bold colours, which draw users’ attention to important information.

GUI Mock-ups

Below are some example mark-up (not final) designs for how the system will look and act.

**Login page**

This page will be used by users to login to the system. An option for new users to sign up will also be available.



**Sign up page**

The sign-up page will be used by new users to sign up. Any disclaimer texts for users signing up will be displayed here. Users will be required to enter their N number so their account information can be linked with their NTU account.

Incentive for the creation of this page was in case a disclaimer was needed to be signed/read before users were able to use the stores.



**Products page**

Once signed in users will be navigated to this page. From here products can be selected/searched for. A menu bar at the bottom of the page consistent across many screens will be available for navigating around the system.



Users, as opposed to staff and admins, have less options and therefore their menu bar will be more simplistic.



**Single product page**

This page will display the necessary information of a product. From here a user can add a product to their basket to checkout. A map of the location of the product within the store will also be available. If no items of the displayed product type are available users will be notified here as to when, if known, the next shipment of this product is expected.

This page will be navigable from various locations within the software in which products are shown.



**Single product page (For staff & admins)**

Admins and staff will have access to more features on this page such as the ones shown below. Blue controls represent ones which only admins will be able to use.



**Check out page**

This will be like a basket page seen on various shopping sites displaying a list of the products which the user wishes to checkout. Users will be able to remove items from this list, check the quality of the items and view their descriptions.



**Check-in page**

On this page staff will be able to search for a customer to see the items they have currently checked out. These items can be selected and checked back in. (This will send them either to the inspection database if it’s developed in time, or flag them as returned in stores)



**Messages page**

The messages page will display messages for the logged in user. Categories of messages include; Overdue returns, products low on stock, expected shipments and expiring products.



**New shipment pages**

This page, used by staff, is to enter information about an arrived shipment. Depending on weather the shipment is a repeat shipment the user will navigate to either the “Add brand new product” page or “Update stock shipment” page.





**Adding new shipments**

The process of adding new shipments is to be split up into two separate pages. First where shipment information is entered, and second where the items of the shipment are entered. These items will then be saved to stores.





**Invoices page**

Two options are available on this page; A staff member can either add an invoice manually to be sent to a department, or, the invoice information can be filled in automatically from the systems invoice log.



**Inspection page**



**3.2 Hardware interfaces**

Thanks to the system being software based there are only two hardware interfaces to be considered. Firstly, the systems interface with the barcode scanner, and secondly the interface between the system and computer on which it will run. Both interfaces will be used by staff, admins and customers.

**3.3 Software interfaces**

The diagram below illustrates the systems software interfaces.

|  |  |
| --- | --- |
| **Interfaces** |  |
| The logical characteristics of each interface between the software and the users have been considered and described |  |
| Interfaces between the system and any external systems have also been described. |  |
| A rough draft of sample screen images have been provided (using Balsamiq or similar). |  |
| Hardware and software interfaces outlined by an appropriate diagram. |  |
| Hardware interfaces could include supported device types etc. |  |
| GUI standards have stated and discussed |  |
| All necessary interfaces have been discussed, and it is made clear that these are not the final screen designs |  |



To transfer data between the database and the system the controller will make use of a database interface class. This will include the retrieval and uploading of; products, user data, transactions and invoices. In order to achieve its functionality, the interface will make use of the internet via PHP API requests.

The systems backend will be linked via an interface to the GUI. This link will provide the user with a method of accessing the systems features.

# Use Case Modelling

## Use Cases

* Provide a summary of the major functions that the product will perform. Organize the functions to be understandable to the customer or a first-time reader. Include use cases and business scenarios or provide a link to a separate document (or documents). A business scenario:
* Describes a significant business need
* Identifies, documents, and ranks the problem that is driving the scenario
* Describes the business and technical environment that will resolve the problem
* States the desired objectives
* Shows the “Actors” and where they fit in the business model
* Is specific, and measurable, and uses clear metrics for success

## Misuse cases

* Provide some examples of functions that should not be possible to perform in the system. These may help you meet any security requirements you have.

|  |  |
| --- | --- |
| **Use Case Modelling** |  |
| Use case diagrams outlining the main use cases for the system. These should be done to the UML standard. |  |
| These should include all the actors mentioned in the introduction and address a problem the system is to solve. |  |
| At least one misuse case documented with comment on how it relates to (for example) security aspects of the system |  |
| Use case diagrams outlining all the use cases for the system. |  |

# Project Plan:

This may be considered as separate from the rest of the document, as it is not concerned with the system, but how you are going to organise your team to build it.

Explain which agile methodology you will use, what tools you will use to track progress, source control etc.

Take a look at your interfaces and assign them “T-shirt sizes” (XS, S, M, L, XL).

* 1. **Approach to development of the system**

The team would start by clearly writing job descriptions for each team member and making them available to every member of the team. Everyone must know their own and fellow members areas of responsibility.

To foster collaboration, the team would implement a specific type of organizational culture - clan culture - which would foster mentorship and team accountability. The teamwork combines individual commitment with group results, which makes it different from regular working groups, such as a function in a corporation. Team members would be assigned roles based on their transferable skills and skill potential, not for their personality (Principles of Management 499).

The team will work their ground rules by discussing expectations of collaboration, discipline, confidentiality matters, project approach, conflict resolution, to name a few. To set tasks and track performance, the team would equip the team members with a methodological framework and versatile tools. The framework would be scrum, one of the implementations of agile methodology. This framework supposes conducting everyday meetings mandatory for all team members where each member reads their tasks for the day, and the status of processed tasks is established. What is more important, everyone thinks of the ways they can help their colleagues succeed in fulfilling the daily tasks. In case the timeline gets challenged, a brainstorm for solutions is conducted. This way, the group would ensure a smooth work process for each member.

The issue of having all team members in one location is easily circumnavigated, since telecommuting is perfectly fine in the contemporary world. Communication would be conducted via emails, messengers, and video conferencing. Emails would be mainly for formal approvals and distributing minutes of meetings, messengers, such as WhatsApp, would serve as daily formal and informal communication channels; and video conferencing would be conducted in applications, such as Microsoft teams or Zoom.

Regarding the tools for building a system, GitHub proves to be an indispensable software development partner within the developers’ community. It encompasses code creation, integration, review, and overall team management. GitHub would also act as storage for other files such as documentation, requirements specifications etc.

Overall, team management is a challenging task. Leaders have to unite team members, supply them with measurable job descriptions, and equip with necessary tools for work. To remain productive, agile methodologies have to be welcomed. The crucial aspect is helping co-workers to reach mutual success.

Below are the assigned project roles based off-of members strengths in subjects which will be used to produce the software.

|  |  |  |
| --- | --- | --- |
| **Member** | **Project related strengths** | **Role** |
| Hamid | Visual studio, SQL, C# experience. | * Co-project leader * Software interfaces programmer |
| Elliot | Visual studio, SQL and PHP experience.  C# with windows forms experience.  Experience with development of similar applications. | * Co-project leader * Software interfaces programmer * UI programmer |
| Haowei | Visual studio, SQL, C# experience | * Database Programmer * Scrum leader |
| Steffan | Visual studio, SQL, C# experience. Proficiency in Photoshop, Illustrator (For UI design). | * UI Designer * UI programmer |
| Michael | Visual studio, SQL and PHP experience. | * Security programmer * Software Interfaces programmer |
| Hamed | Visual studio & SQL experience. | * Database Programmer * Security programmer |

* 1. **Backlog**

To track working time on each task, as well as their status, the team would introduce Jira or Trello. New assignments added to a backlog would then be moved to the stages, in progress, followed by approval, revision if necessary, and completion. These digital logs provide managers and employees with statistics regarding the productive time one works and allow predicting timeframes for upcoming projects.

Below is the initial backlog of the project:

// todo finish this off

* Stock management software – Epic
  + User interface
    - Base user interface class
    - All UI page classes
  + Controller
    - Page manager
    - General functionality
  + Database
    - Table design
    - Table implementation
    - Filling with test data
  + Additional classes
    - Tools class
      * Email functions
    - Database classes
    - Controller-Database interface class
  + Testing
    - UI testing
    - Test framework creation

|  |  |
| --- | --- |
| **Project Planning** |  |
| The agile methodology that will be used has been clearly stated and explained |  |
| The benefits and potential drawbacks of the agile methodology chosen have been discussed. |  |
| Details of the end of sprint meetings have been provided |  |
| The role each member of the team will take on has been stated. |  |
| Each members strengths have been briefly discussed |  |
| Evidence of an agile estimating technique such as function point analysis, T-shirt sizing, Story points or use case points. |  |
| A backlog has been provided. |  |

References

## 

References here in Harvard referencing style. Use MS Word citing and referencing feature.

*Principles of Management*. Openstax. Roice University. (Came from project planning section, needs review)

Bibliography

## 

Bibliography here in Harvard referencing style.

(S Dhall & P Sundararaman, NA). C# AND .NET FRAMEWORK. <https://www.cs.colorado.edu/~kena/classes/5448/f12/presentation-materials/dhall.pdf>