Software Requirements Specification

**Version: Draft**

**October 09, 2019**

**Revision Chart**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Primary Author(s)** | **Description of Version** | **Date Completed** |
| Draft | All team members | Initial draft created for distribution and review comments | October 09, 2019 |
| Preliminary | TBD | Second draft incorporating initial review comments, distributed for final review | TBD |
| Final | TBD | First complete draft, which is placed under change control | TBD |
| Revision 1 | TBD | Revised draft, revised according to the change control process and maintained under change control | TBD |
| Revision 2 | TBD | Revised draft, revised according to the change control process and maintained under change control | TBD |
| etc. | TBD | TBD | TBD |

**Contents**

**1.** **Introduction**

1.1 Purpose

1.2 Scope

1.3 Definitions, Acronyms, and Abbreviations

1.4 References

**2.** **Overall Description**

2.1 Product Perspective

*2.1.1* *System Interfaces*

*2.1.2* *User Interfaces*

*2.1.3* *Hardware Interfaces*

*2.1.4* *Software Interfaces*

*2.1.5* *Communications Interfaces*

*2.1.6* *Memory Constraints*

*2.1.7* *Site Adaptation Requirements*

2.2 User Characteristics

2.3 Constraints

2.4 Assumptions and Dependencies

**3.** **Specific Requirements**

3.1 Software Product Features

*3.2.1* *Feature 1*

3.2 Performance Requirements

3.3 Software System Attributes

*3.3.1* *Reliability*

*3.3.2* *Availability*

*3.3.3* *Security*

*3.3.4* *Maintainability*

*3.3.5* *Portability*

3.4 Logical Database Requirements

3.5 Other Requirements

**4.** **Appendices**

# **1.** **Introduction**

## **1.1** **Purpose**

The purpose of this specification is to provide assistance for a freight forwarding company, thus our audience. First of all, the specification will help manage and process all the data of the clients, the shipment, the arrivals and departures of the stocks. Moreover, it will help in the organization and management of the warehouse.

## **1.2** **Scope**

There will be one software product, the Warehouse Manager. The Warehouse Manager will do the below functions.

* Register a new client to the Database
* Edit the information of an existing client
* Delete a client
* Show the transfer history between the client and the company
* Calculate and print the receipt to the client
* Provide authorized authentication using username and password for the staff. (Different users will have different permissions.)
* Import shipment to the warehouse
* Export shipment from the warehouse
* Organise the warehouse
* Print the proper documents for importing and exporting
* Import information to database using scanner device (?)
* Notify storekeeper if “food stuff” are going to expire soon
* Choose which “food stuff” will leave first (FIFO)
* Inform where is the location of a specific shipment in the warehouse
* Inform where we have free space in warehouse
* Indicate where is best suited to store a shipment
* Notify storekeeper if warehouse is getting full
* Notify storekeeper for packages that stayed too long in the warehouse

Warehouse Manager will not provide:

* Fully automated system. Humans will be a part of the system.
* Access to the clients. Only for staff

Benefits

The benefits are that by using Warehouse Manager the company will save a lot of time and the load of work will be reduced drastically. Moreover, there will be more security since there will be a login process now. Another benefit is that the storekeeper does not need to check the shipments daily, as the product will provide this information to him. Also we will maximize the usage of the warehouse so no space will be left unexploited.

Objectives and Goals

Provide better support to the clients and organise efficiently the data and warehouse.

## **1.3** **Definitions, Acronyms, and Abbreviations**

Pallets = Is a close box containing the shipment. One transfer can have a lot of pallets.

food stuff = Items that have an expiration date. Usually is food or drinks but it can be other things too. E.g. medicine

## **1.4** **References**

No references in this document

# **2.** **Overall Description**

## 

## **2.1** **Product Perspective**

The product will be independent and will not communicate with other products.

The various parts of the program will only communicate with each other.

### **2.1.1** **System Interfaces**

The system enables the user to search for information about the products in the warehouse (Database). Also, specific users have the ability to add, remove or edit information about products in the warehouse (Database).

### **2.1.2** **User Interfaces**

Users of the product have little knowledge of this type of systems, so the user's contacts with the system should be simple, short and self-instructive.

The data will be in table format with different fields. The user will be able to sort the items by field.

There will be a search menu where the user can search for information based on a field.

For example:

The field is the name and in the box the user writes the desired name.

The field is the product type and the box writes the item.

It will also be possible to search with multiple fields.

There will be a method of adding new entries to the table. There will be a form which requests the record’s information and will explain each field briefly. There will also be error messages. For example, error messages for 2 records in the same location.

There will be a method to delete records from the table which will ask the user twice before deleting.

There will be a method of changing a record that is similar to the input method.

There will be a log-file showing all operations within the system.

**2.1.3** **Hardware Interfaces**

There are no special hardware requirements, the product must be capable of running on most computers running Windows as an operating system.

### **2.1.4** **Software Interfaces**

Name: Microsoft SQL Server 2017 Express

Mnemonic:

Specification Number:

Version Number: 14.0.1000.169

Source:<https://www.microsoft.com/en-us/download/details.aspx?id=55994>

Name: SQL Server Management Studio

Mnemonic:

Specification Number:

Version Number: 18.2

Source:<https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-2017>

Microsoft SQL Server 2017 Express και SQL Server Management Studio :

Will be used for the creation and management of DB.

Name: Java SE Development Kit 8

Mnemonic:

Specification Number:

Version Number:8u221

Source:<https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

### **2.1.5** **Communications Interfaces**

The application will need to communicate with its own database using the internet.

### 

### **2.1.6** **Memory Constraints**

There are not any specific constraints , the program must be able to run on most modern computers .The database must have its own computer with its own amount of storage.

### **2.1.7** **Site Adaptation Requirements**

\*\*\*\*\*There’s no site adaptation requirements

## **2.2** **User Characteristics**

No specific educational level is required to use this application.

Computer experience is a benefit but not required.

Warehouse/Inventory management experience is required.

## **2.3** **Constraints**

Budget.

This is a project for a university assignment. Despite having a real client it’s not meant to be fundable.

Access to the clients database is not allowed.

Hardware limitations:

The use of external hardware such as scanners along with bar codes will produce a more efficient and productive application but unfortunately it’s not feasible.

There is not unlimited storage, the database must delete old files to save storage.

## **2.4** **Assumptions and Dependencies**

Increased warehouse storage size

Bigger warehouse means more inventory to manage. A potential upgrade to the warehouse may change some requirements such as memory or hard disk space.

New products can have an impact, especially if they need special conditions to be stored, shipped etc.

If the client decides on providing new services, some extra requirements may arise.

**3.** **Specific Requirements**

1. System requires a Database to keep the list of the clients of the company. The DB will also have every object recorded and sorted by kınd, owner and client. The DB user will be able to add , remove or edit any clients and objects. Also there will be a log file where every change will be recorded for security purposes.(With the scanner device any scanned barcode will be imported into the system to minimize human error and interaction in importing data to the database.)
2. The system will keep track of the transfer history between any client and the company and the admin will be able to recall and see the past transfers.
3. The system will be able to add the sum of the transactions between the clients, remove the expenses, calculate the total amount and give a receipt to the client.
4. The system will also have an integrated login system that will only let employees, admins and authorized people to sign in and use it.
5. The system will inform the admin with alerts when shipments is received or shipment left the warehouse.
6. The user will be able to see the organization of the warehouse at any given time and where there is free space or occupied space.
7. The system will notify with alert whether the food is going to expire soon.
8. There will be a priority queue in the system that will decide which item must leave first and which last, this function will work only for food stock.
9. The system will help in finding wanted items fast by searching them with name of product, owner, kind of product or date received.
10. The system will inform the admin about the capacity of the warehouse and when it goes close to maximum or minimum.
11. The system will be sorting the items by size,owner,fragile,kind of item into the warehouse.

## **3.1** **Software Product Features**

#### **3.1.1 Feature 1:**

The database is mainly needed to keep track of the objects that are being stored in the warehouse. As a user of the system, someone can see the objects stored, add new objects, modify the already stored items, remove objects that have been delivered to the client and as well see the items sorted by kind or by owner/client . Also, the database will store all the clients that the company serves or served in the past.

Introduction :

manageDB(action )

This is basically the core function . It gives access (full or partial depending on the credentials that the user has entered ) to the database . We suppose that the user is someone with authorized access . Through the specified input , he can delete , enter or edit the contents of the database . He is allowed to make changes on both clients and objects . Note that there is going to be a log file that will store every change occurred in the system ( like the systems they use in banks . ) .

Inputs :

The input is a command like : insert , delete , edit . Then it should be followed by a token to identify whether we are talking for a client or an object . After that , this function will call another smaller more specific function , that will ask for more info about the action wanted . Like what changes does the user want to make . This info for e.g could be : add a client , add an object , edit information etc .

Processing :

As for the processing part , we know that the system and the database are always connected and interfering with one another . Therefore the process will be done by the database as soon as the user wants to . If the request does not get past fault controls , the DB will return an error message and prompt the user to re-enter it . An addition is the log file , which can be something as simple as a text file and it will contain everything that anyone does in the system

Outputs:

The output is just the change in the database represented visually . If the user adds a new client , all the DB users will be able to instantly see the new addition to the database . Same stands for every action .

**3.1.2-3 Features 2 & 3:**

The system will keep track of the transfer history between any client and the company and the admin will be able to recall and see the past transfers.

The system will be able to add the sum of the transactions between the clients, remove the expenses, calculate the total amount and give a receipt to the client.

Introduction :calculate receipt( client , Trac\_info\*) {

As soon as a transaction occurs , the system is informed . The user will simply enter the information about the transaction . Then , the required changes will take place in the database ( inform objects and clients list ) using the function above . The company’s logistics are instantly updated with the transaction details . Using the function calculateReceipt(client ) a receipt will be ready in almost no time. Note that every transaction is saved as well as every receipt .

}

viewTransactionHistory ( transactionLog\*){

For the second part , when a user has signed in , using admin credentials , he will have an extra choice to view the transaction history . He can see the transactions in any way he wants ( date , client , amount ) as we store every transaction .

}

Inputs : client - all information needed for a client ( name , ID , company , etc )

Trac\_info\*- all information for the specific transaction ( serial number , amount , date , quantity , sale , etc )

transactionLog\*- could be anything from a specific date , a date range , a client , a product . Generally the admin can see any transaction about everything in the company in the way he wants it .

Processing : As mentioned above , transactions are semi-automatic . This means that the user has to enter the info , and then the system will do simple things such as mathematic calculations .

Outputs : A receipt will all the necessary info needed on it .

The log of all the transactions .

**3.1.4 Feature 4**

The system will also have an integrated login system that will only let employees, admins and authorized people to sign in and use it.

Introduction : void authLogin ( username , password ) {

Self explanatory function .

**3.1.5 Feature 5:**

The system will inform the admin with alerts when shipments is received or shipment left the warehouse.

Introduction: string informMsg (database) {  
Whenever there is a change in the quantity of products in the warehouse ( addition , extraction ) the administrator is informed either via telephone message or e-mail .

This function interferes with function number 1 , which is responsible for adding /removing objects from the warehouse

}

Processing : We will use what are called “triggers “ to inform the admin about the condition of stock in the warehouse .

Output : A message containing information about arrival or departure of stock .

**3.1.6 Feature 6:**

The user will be able to see the organization of the warehouse at any given time and where there is free space or occupied space.

Introduction : warehouseState ( warehouse,database) {  
Gets information about the state of the warehouse . In the database , we know the exact place that each object is stored on . This function will use that information to display visually using GUI where exactly is every object in the warehouse .

}

Inputs : The exact location of every object . Can be found from the database .

Processing : Generate a 3D model that is easy to navigate through .

Output : The 3D representation of the warehouse .

**3.1.7-8 Features 7&8:**

The system will notify with alert whether the food is going to expire soon.There will be a priority queue in the system that will decide which item must leave first and which last, this function will work only for food stock.

Introduction : foodStock ( database->select from food -> expiry date less than X days ) {

For every object considered as “food “ we use some sort of priority queue to ensure that the one that expires the soonest is the one to leave the warehouse first . Once a day or when the user wants to , the system automatically looks up to the database to see whether there is “food “ that needs to leave the warehouse . If it notices an expiry date sooner than X days , the user is informed with an alert message .

}

Inputs : every object that belongs to the category : food

Processing : Check the expiry date , if it needs to be reported , do so .

Output : Alert message , priority queue .

**3.1.2 Feature9:**

The system will help in finding wanted items fast by searching them with name of product, owner, kind of product or date received.

Introduction : findItem ( info ,database) {

User can find a specific object either by performing a search via name , via owner , via date etc .

Inputs : Info - The key for the search

Processing : locate the requested object(s) and all of it’s info .

Outputs : Info of the object .

Function 10 ) The system will inform the admin about the capacity of the warehouse and when it goes close to maximum or minimum

Introduction : capacityMessage ( warehouse.capacityLevel ) {

Pretty much self explanatory

}

## **3.2** **Performance Requirements**

The number of simultaneous users the system supports is more than 10 , but the system will only be used at any given time by maximum 2 people. One is the admin that has the rights to check or change anything he likes and the user which is the storekeeper who will be adding the data by scanning barcodes and adding the items in the correct positions in the warehouse.

The amount of data the system can process is one barcode at a time. The input will be added one by one with the help of the scanner to minimize input error. Although the system will need to add one by one the codes , the process will be very fast because it doesn’t need a lot of effort to scan a barcode.

## **3.3** **Software System Attributes**

### **3.3.1** **Reliability**

Our software will be failure-free and whatever problems may occur it is guaranteed that in the small period of 1 week they will be fixed. Our software will stay at the same position as when it was created unless there are changes caused by the hardware.

### **3.3.2** **Availability**

Our software will be highly available most of the time. It will function non-stop from the day installed on the system until the user decides to change or update the software. To guarantee a certain level of availability the system must not be changed or updated by other than the creators, except if the right licenses and permissions are given to a person. The system must be protected so no unauthorized person has access to it because it is designed to be controlled, only be staff, admins and trusted members.

### **3.3.3** **Security**

The security measures we took for our software are activity logging for example anyone that ever logged into the system and whatever they changed will be saved in a log file for debugging or for the admin to be able to identify any problems that may occur by outsiders. There will also be an admin login and a user login that will be used accordingly and the user privileges will be able to edit anytime by the admin of the software. The data will be checked and validated by the admin and because every input will be automated ,with limited human interaction, the data will be correct . There are no other restrictions at the moment.

### **3.3.4** **Maintainability**

We will maintain to provide support for the system after it is delivered and we will fix any bugs that may occur. Our software is not very complicated or very simple but with the simplicity of the structure that we will design the software it will be easy to debug and fix any problems.

### 

### **3.3.5** **Portability**

The system will not be portable it will be installed on a number of devices like the company owners and the warehouse storekeepers and only a few people will have access on those devices. It will stay at the company and only there someone will be able to access the system.

## **3.4** **Logical Database Requirements**

Not available yet.

## **3.5** **Other Requirements**

**4.** **Appendices**

*Include supporting detail that would be too distracting to include in the main body of the document.*