Software Requirements Specification

for

MPSS - Motor Part Shop Software

Version 1.0 approved

Prepared by,
Swapnil Yasasvi (20CS30054)
Kulkarni Pranav Suryakant (20CS30029)
Sidharth Vishwakarma (20CS10082)

Indian Institute of Technology Technology, Kharagpur

16th March, 2022

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

The document's goal is to serve as a reference for designers, developers, and testers working on the ***Motor Part Shop*** *project's engineering. It should include all of the information required for the engineers to design, create, and test the programme.*

1.2 Document Conventions

In general, this document adheres to IEEE formatting standards. The text in the document is single spaced with 1" margins.

Conventions	Description	Examples
Font (Times New Roman)	*Whole of the document will be written in this font only.*	
Bold	*Bold face and indentation are used on general topics and or specific points of interest. Headings of the topics are written in Bold fonts. Web site links.*	**1.2 Document Conventions**
Italics	*personalized comments and notes*	
Capital letters	*Shorts forms of words*	**SRS, MPSS**

1.3 Intended Audience and Reading Suggestions

Developers, shop owners, testers, and documentation writers should read this page.

Begin with the goal, then move on to the product scope, the system's functional and non-functional requirements, and the contextual and data flow diagrams.

1.4 Product Scope

A modest auto parts store supplies replacement components for a variety of car types. In addition, each component is often produced by a number of small businesses. The shop owner has asked us to design the following motor component shop software to streamline sales and supply ordering.

The motor part business handles a huge number of motor components from a variety of manufacturers and vehicle kinds. Some of the motor parts are quite microscopic, while others are extremely massive. Different parts are kept in wall-mounted and numbered racks by the owner.

After being influenced by the "just in time (JIT) attitude," the store owner keeps as little inventory for each item as possible to cut inventory overheads.

The system can be utilized by:

- ➢ *Shop owner (with unlimited access)*

1.5 References

- ➢ *IEEE Standard 830 - 1998 IEEE Recommended Practice for Software Requirements Specifications, IEEE Computer Society, 1998.*
- ➢ *Slides from the NPTEL course Object Oriented Analaysis and Design by Prof. Partha Pratim Das, IIT Kharagpur.*

2. Overall Description

2.1 Product Perspective

The Motor Part Shop system will be a brand-new, self-contained system. This part contains the contextual diagram, which contains a wealth of information as well as the interactions between various entities.

2.2 Product Functions

The software for a motor parts store takes care of the following:

- ➢ *A large number of motor components from various manufacturers and vehicle kinds are available.*
- ➢ *Different parts are kept in wall-mounted and numbered racks by the owner.*
- ➢ *After being influenced by the "just in time (JIT) attitude," the store owner keeps as little inventory for each item as possible to cut inventory overheads.*
- ➢ *The solution solves the shop owner's problem of being able to order things as soon as the number of items in the inventory drops below a certain threshold value, and it allows the shop owner to keep components in stock for around a week of sales.*
- ➢ *The product allows the user to produce the products to be ordered, print out the part number, the needed quantity, and the vendor's address.*
- ➢ *It also allows the firm to gain income on a daily basis as well as at the end of the month. The programme creates a graph that displays the sales for each day of the month.*

2.3 User Classes and Characteristics

The following are some of the consumers that are looking forward to using this product:

Owner:

- ***Maintenance:*** *The racks are numbered and put on the wall. Streamline sales and supply ordering while adhering to the JIT principle.*
- ***View the following information about the customer:*** *View the customer's personal information.*
- ***Managing Customer Sales:*** *Responsible for correctly assigning and delivering the selected product to the client based on the customer's preferences.*
- ***View Product Stocks:*** *For selling purposes, keep track of each product item's stock.*

2.4 Operating Environment

The software will run on a Linux computer system.

2.5 Design and Implementation Constraints

The following are the system's design and implementation constraints:

- ➢ *Python and SQL should be in charge of the back end.*

2.6 User Documentation

A README.md file will be included with the software, which will serve as the user manual. It can also be printed out and presented to the user. It will include all of the essential instructions for setting up and using the programme so that you can take advantage of all of its features. The user manual (README file) will be created in accordance with industry best practises and norms.

2.7 Assumptions and Dependencies

There is no need to create additional owner accounts since the program expects that only one owner would use it. It also implies that just one item is entered when registering a sale (quantity might be many), and if multiple types of items are sold, the owner can report them as distinct sales multiple times. Another assumption made by this programme is that at the conclusion of each day, the owner prepares a list of products to be ordered. Furthermore, the proprietor consistently orders the recommended number of these things. Aside from that, the software just requires a functional computer, installed dependencies, and sufficient storage space for the data.

3. External Interface Requirements

3.1 User Interfaces

The primary screen appears when the software is started. After that, the user may decide what he wants to do with it. It has a tab for selling products to the buyer, where the customer may choose which items to buy and for how much. An invoice is generated at the conclusion. Another tab allows the store owner to identify whether goods in the inventory have remaining stock levels that are fewer than the required threshold. The business owner may then place orders for these things from the suppliers on the other tab based on his needs. The last tab will assist the owner in analyzing his sales by calculating the revenue made for the day and month. It also includes a graph depicting sales by day of the month. The user interface has been kept simple and straightforward for ease of usage.

3.2 Hardware Interfaces

The desktop/laptop on which the programme will operate is the sole hardware component used. A good processor will help to speed up database processing and queries. Aside from that, no unique hardware interactions or components are present.

3.3 Software Interfaces

Using the sqlite3 package in Python, the data from the database may be retrieved directly from the application. The programme does not require the usage of a web browser to work. Python's Electron module was used to create the GUI design. The entire software is developed in the Python programming language.

3.4 Communications Interfaces

There will be no need for an internet connection because this is not an online programme. The programme extracts data from the system's database files and saves the changed data in the same database file for future use.

4. System Features

4.1 Displaying Sales Statistics

4.1.1 Description and Priority

This function assists the owner in doing numerous analytics activities such as calculating income at the conclusion of each day and month, as well as displaying a graph indicating sales for each day of the month. This may be thought of as a bonus feature.

4.1.2 Stimulus/Response Sequences

After the order list has been created at the end of each day, the owner will have the option of viewing the income made that day. The relevant income will be presented when you pick that choice. There will also be an option to construct a graph displaying sales for each day of the preceding month, which will be presented if the proper choice is selected.

4.1.3 Functional Requirements

REQ-1: The programme must be able to save income for each day of the previous month.

REQ-2: The programme should be able to graphically display daily sales for a month on the screen.

4.2 The threshold is displayed, as well as a list of parts that need to be ordered

4.2.1 Description and Priority

This feature allows the owner to produce a list of all the things that have gone below their threshold value and need to be ordered at the end of the day. This is the most important function since it is the task that, when automated, saves a lot of time and energy.

4.2.2 Stimulus/Response Sequences

The owner will choose a button to signify that the current day has finished to signal that a certain transaction was the last sale of the day. After that, the programme will automatically determine which goods are in short supply (have gone below the quantity threshold) and must be purchased.

It will create a table with a list of similar objects, and the owner will have the choice to save the list.

4.2.3 Functional Requirements

REQ-1: At the end of the day, the programme should be able to compute the threshold for each item by computing the average sale of that item.

REQ-2: The programme should display the list of products to be ordered in a tabular manner, together with their quantity and vendor address, and provide the user the opportunity to save the list.

4.3 Purchasing from vendors, choosing from a variety of vendors

4.3.1 Description and Priority

The business owner orders the things that fall below the threshold value at the end of the day. For this, the store owner first chooses the vendor from whom he wants to buy (if there are numerous vendors, choose one from the list) and then puts the order.

4.3.2 Stimulus/Response Sequences

On his or her home screen, the owner will have the choice to delete a specific item from the inventory. It will be deleted after choosing the item and confirming that it is existent in the inventory database.

4.3.3 Functional Requirements

REQ-1: The programme should be able to get the list of products that are less than threshold and their corresponding vendors.

REQ-2: The programme should display the list of products to be ordered in a tabular manner, together with their quantity and vendors' information listed to choose.

4.4 Selling to customers

4.4.1 Description and Priority

This feature allows the owner to keep track of the specifics of any transaction made during the day.

This function makes adjusting the number of each item in the inventory a lot easier, therefore it's a really useful tool.

4.4.2 Stimulus/Response Sequences

On the home screen, the owner will have the opportunity to register a transaction. He or she will then be sent to a separate page where they must pick the item that is being sold. The owner will first be prompted to select an item type from a drop-down menu. Then, in a new drop-down menu, you'll see a list of all the manufacturers who create that item. Following the selection of a certain manufacturer, a new drop-down menu will appear, displaying the vehicle kinds available for the specified item category and manufacturer. The item selection procedure will be completed once you have chosen a vehicle type. In addition, the seller must provide the amount of the item being sold. The sale will be executed after confirmation, and the inventory information will be correctly updated.

4.4.3 Functional Requirements

REQ-1: The items showing in the next drop-down menu should vary dynamically depending on the item selected in the previous drop-down menu in the three-level drop-down list.

REQ-2: The programme should be able to verify if the sold quantity is positive.

REQ-3: The programme should be able to update the inventory database with the quantity of the item sold.

4.5 New parts are being added

4.5.1 Description and Priority

The owner can use this function to add a new item to the inventory. Although this feature isn't the most important, it aids the operation of a real-world vehicle spare parts business because new things, parts, and tools enter the market on a regular basis.

4.5.2 Stimulus/Response Sequences

After entering into the system, the owner will choose the option to add a new item, which will prompt the system to ask for data in fields such as item type, manufacturer, vehicle type, price, and beginning quantity. The new item will be uploaded to the inventory database when this information has been confirmed, and a success message will be shown.

4.5.3 Functional Requirements

REQ-1: The programme should be able to do sanity checks to make sure the item isn't already in the inventory database.

REQ-2: It should be able to detect any data fields that have been left blank, as well as verify that the amount of the item submitted is positive.

REQ-3: The item's information should be added to the inventory database by the programme.

4.6 New parts are being removed

4.6.1 Description and Priority

This feature enables the user to delete an item's existence from the inventory database.

This isn't a must-have feature, but it aids in simulating the routines and operations of a real-world auto parts store.

4.6.2 Stimulus/Response Sequences

After entering into the system, the owner will choose the option to add a new item, which will prompt the system to ask for data in fields such as item type, manufacturer, vehicle type, price, and beginning quantity. The new item will be uploaded to the inventory database when this information has been confirmed, and a success message will be shown.

4.6.3 Functional Requirements

REQ-1: The programme should be able to determine whether or not the item being withdrawn is in stock.

REQ-2: The programme should be able to erase the item's record from the database, as well as, if necessary, the manufacturer's entry providing that item, assuming that the same manufacturer does not provide any additional items.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The programme is an offline application that works without the need for an internet connection, thus it will be available 24 hours a day, seven days a week.

5.2 Safety Requirements

The program runs on top of numerous abstraction layers on top of the hardware, with low risk of harm to the owner's device. If the owner suffers any business losses, however, he or she is responsible.

5.3 Security Requirements

When the programme is loaded, it shows a login screen to prevent anyone other than the store owner from using it. The owner may only proceed to the next page after entering the right login and password. From there, he or she can access the inventory or make any modifications to the data.

5.4 Software Quality Attributes

The programme should be simple to use and maintain. It will be able to adapt to new features with little code modifications. Upgrading to newer versions of the programme will be simple. This will improve the software's maintainability and reusability. Once launched, the programme will allow for input in order to include any future requirements. The programme would be adaptable to any third-party software with which it may interface.

The user interface will be welcoming, making it simple to comprehend the features and how to utilize them.

5.5 Business Rules

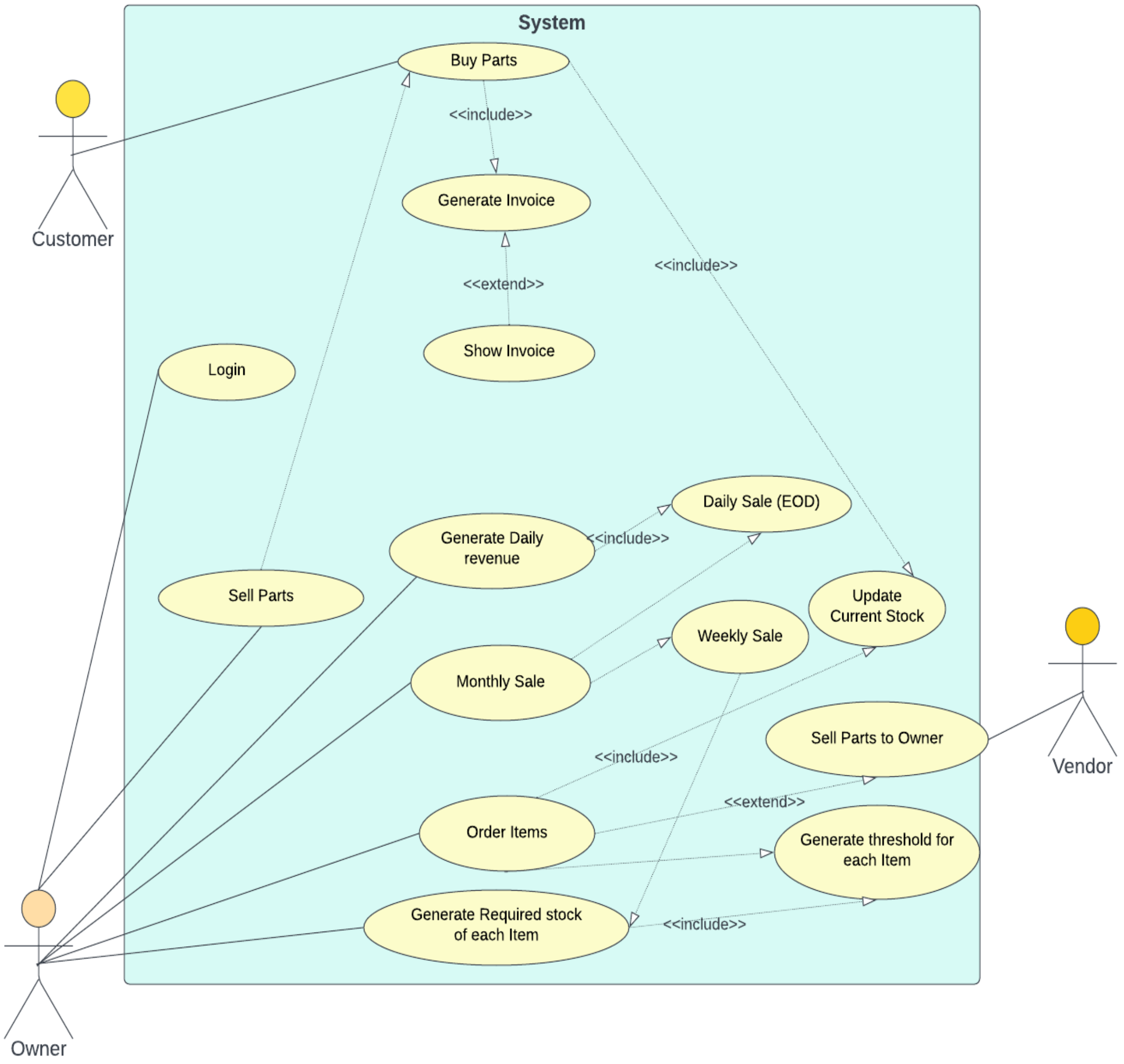
Because he is supposed to be the single user, the store owner is able to utilize all of the software's functions. Also, without prior approval, the programme should not be outsourced to any third party. Before any industrial usage, the project holder reserves all applicable rights of the project licenses and permit

Appendix A: Glossary

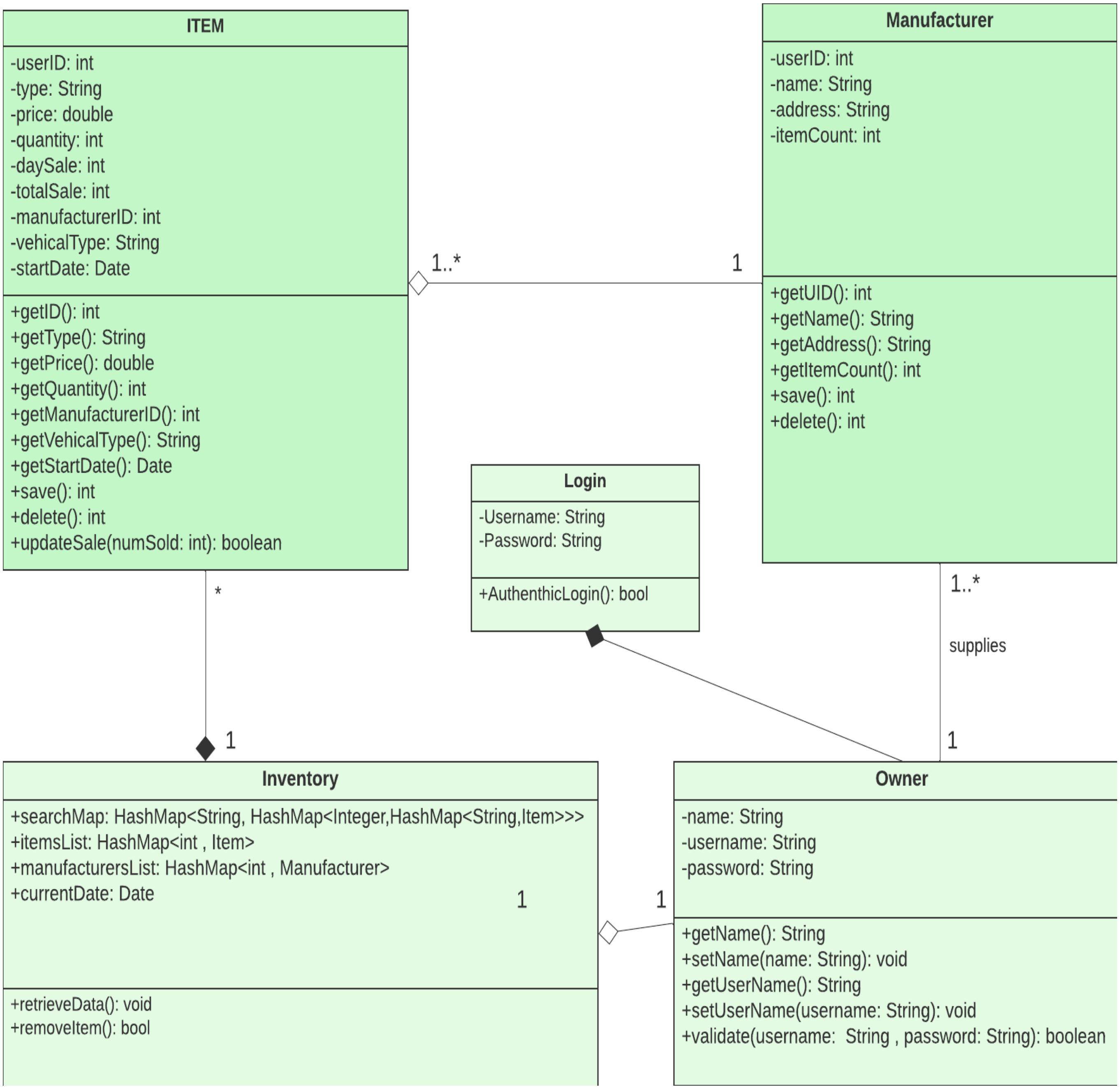
Acronym, Abbreviations	Description
SRS	*Software Requirements Specifications*
IEEE	*Institute of Electrical and Electronics Engineers*
SQL	*Structured Query Language*
GUI	*Graphic User Interface*

Appendix B: Analysis Models

1. Use - Case Diagram



2. Class Diagram


ITEM
-userID: int
-type: String
-price: double
-quantity: int
-daySale: int
-totalSale: int
-manufacturerID: int
-vehicalType: String
-startDate: Date
+getID(): int
+getType(): String
+getPrice(): double
+getQuantity(): int
+getManufacturerID(): int
+getVehicalType(): String
+getStartDate(): Date
+save(): int
+delete(): int
+updateSale(numSold: int): boolean
1..*
1
Manufacturer
-userID: int
-name: String
-address: String
-itemCount: int
+getUID(): int
+getName(): String
+getAddress(): String
+getItemCount(): int
+save(): int
+delete(): int
Login
-Username: String
-Password: String
+AuthenthicLogin(): bool
*
1
1..*
supplies
1
Inventory
+searchMap: HashMap<String, HashMap<Integer,HashMap<String,Item>>>
+itemsList: HashMap<int , Item>
+manufacturersList: HashMap<int , Manufacturer>
+currentDate: Date
1
1
+retrieveData(): void
+removeItem(): bool
Owner
-name: String
-username: String
-password: String
+getName(): String
+setName(name: String): void
+getUserName(): String
+setUserName(username: String): void
+validate(username: String , password: String): boolean