STOCK MAINTENANCE SYSTEM

PROBLEM STATEMENT:

The stock maintenance system is all about maintaining the details of goods.In shops,the shopkeeper has to maintain the details of purchase,sales,profits,loss of all the products available.This process was done manually in olden days.But now,as the products are increasing,it is becoming very difficult to maintain all the details of the products in given time.It requires man power and is also a less efficient process.It may also result in errors very often.So,to overcome this problem,people came up with the idea of developing a software which contains all the details of products,such as product ID,quantity,price and many more.There are also modules like sales details,purchase details,profit details and loss details.All these are stored in a database and are automatically generated when the product ID is given.

This idea of building a software to maintain the details of the stock is called “STOCK MAINTENANCE SYSTEM”.It’s a highly efficient and useful software that’s going to serve our purpose.

The main objective of this process is to overcome the work load and time consumption which makes the maintenance of stock in an organization as a tedious process.This project provides complete information about the details of stock to the users.This project identifies the amount of stock available,the products that are purchased,faults in the product , products that are replaced,products that have been sold and the date at which the products are brought and sold in particular concern.

**TABLE OF CONTENTS FOR SOFTWARE REQUIREMENTS SPECIFICATION:**

|  |  |
| --- | --- |
| CONTENTS | PAGE NO |
| Introduction | 5 |
| Purpose | 8 |
| Project Scope | 9 |
| References | 10 |
| Product Perspective | 10 |
| Tools to be used | 10 |
| Definitions | 11 |
| User Characteristics | 12 |
| Functional Requirements | 12 |
| Non Functional Requirements | 14 |
| Hardware Reqs | 16 |

SOFTWARE REQUIREMENT SPECIFICATION:

INTRODUCTION:

The main objective of this project is to maintain the details of stocks sold and purchased.All these details are stored in a data base.As soon as the customer purchases a product,the product ID is entered and the sales details are updated.Then,automatically the software updates the profit or loss details of the product,stores the date of purchase,quantity,price,etc.The purpose of this software is to reduce the inefficiency,decrease the dependency on man power and reduce the time for calculation and updation of details. It is user friendly,helps us save our time and increases the efficiency of the process.

It basically consists of four modules:

1.Item details

2.Purchase details

3.Sales details

4.Supplier details

1.Item details:

The item Module deals with information about the details of the product that the concern handling. It consist of details such as the name of the product, id generated, quantity, cost, etc. This information is retrieved during the sales and purchase of a product.

2.Purchase details:

The purchase Module deals with various item name, supplier name, quantity of the product,color of the product, amount per unit etc.

3.Sales details:

The Sales Module maintains daily sales of the product of the organization. It consist of details such as the item name, sales id, quantity, amount etc. This information is retrieved from the database when a sales detail report or view activity is made by the concern.

4.Supplier details:

The Supplier Module deals with the information about the details of the suppliers giving product to the organization. It consist of details such as supplier name, address, email id, Phone number, sales tax number etc. This information is retrieved when a purchase is done.

PURPOSE:

The main **goal** of this project is to reduce manual works, increase the processing speed and ensure reliability of data. This project “**Stock Maintenance System**” deals with stock Management in a concern. This project also takes care of maintaining records of all the sales details, purchase details of the concern.

It is a very useful software and helps in reducing the man power.The chances of commencement of errors is very less and thus is preferable over the manual process.

PROJECT SCOPE:

In this project,we are going to acheive a software that computes the sales details,purchase details,profit,loss etc automatically.As soon as the product ID is entered,the product details should be displayed.It consists of fields such as name of the product,price,quantity,weight,etc..In similar way several other modules can also be displayed. Thus the main scope of this software would be to maintain different modules such as sales modules,purchase modules,product module etc and upgrade the database accordingly.

REFERENCES:

IEEE Software Requirement Specification format.

PRODUCT PERSPECTIVE:

The Stock maintenance acts as an interface between the 'customer' and the 'salesperson'. This system tries to make the interface as simple as possible and at the same time not risking the work of data stored .

TOOLS TO BE USED:

• Eclipse IDE (Integrated Development Environment)

• Rational Rose tool (for developing UML Patterns)

DEFINITIONS:

* HTML-Hyper Text Markup Language
* HTTP-Hyper Text Transfer Protocol
* TCP/IP-Transmission Control Protocol/Internet Protocol is the communication protocol used to connect hosts on internet.
* J2EE-programing platform for developing distribute java applications
* SQL-Structured Query Language

USER CHARACTERISTICS:

SUPPLIER:

He/She is the person who supplies goods or products to several shops,markets and administrator.

ADMINISTRATOR:

He/She collects goods from sales person and maintain a record of sales details ,purchase details,profit and loss details in respective databases.

SALES MAN:

He/She acts as a supplier of goods between administrator and customer.This actor is not mandatory.

CUSTOMER:

He/She buys the goods from administrator directly or indirectly. The details of the product and customer are stored in a data base and are accessed when required.

FUNCTIONAL REQUIREMENTS:

1.Login: Login is achieved by the stock manager.

2.Analysis of Goods:

Finding the expired goods

Finding the older ones and selling with offer prices

3.Preparing the List: List of goods or items which are needed are prepared by the stock manager.

4.Getting the Quotations: Stock manager gets the quotation from the company manager.

5.Choosing the best one: Stock manager chooses the best quotations.

6.Purchasing the goods: Stock manager purchase the good from the company manager.

7.Delivery & Payment: Delivery of goods by the required company and payment settled by the stock manager.

8.Update: It is performed by the stock manager in the database.

NON-FUNCTIONAL REQUIREMENTS:

i. PERFORMANCE REQUIREMENTS:

Performance requirements define acceptable response times for system

functionality.

The load time for user interface screens will take no longer than two

seconds.

The login information shall be verified within five seconds.

Queries shall results within five seconds.

ii. DESIGN CONSTRAINTS:

The software shall be a standard system running in a windows environment. The

system shall be developed using rational enterprise suite.

iii. RELIABILITY:

Specify the factors required to establish the required reliability of the software

system at time of delivery.

iv. AVAILABILITY:

The system should have an availability of 99.99%.

v. PORTABILITY:

The system should be extremely via the usb drive.

The system shall be easy to migrate or backed up via another use drive.

vi. MAINTAINABILITY:

The system shall utilize interchangeable plugins.

The system shall be easily updatable for fixes and patches.

The system shall be easy to upgrade.

HARDWARE REQUIREMENTS:

1. Processor – Intel Pentium IV-2.0 GHZ.

2. Hardware – 40 GB.

3. RAM – 512mb

4. DVD RAM – 1 nos.

SOFTWARE REQUIREMENTS:

1. OS – windows XP/vista.

2. Front end Tool – Rational Rose Enterprise suite.

3. Back end Tool – Oracle 10i.

SOFTWARE CONFIGURATION MANAGEMENT:

Software Configuration Management is the arrangement of exercises which control the changes in the document and identifying the items for change. If something goes wrong, SCM can determine what was changed and who changed it. If a configuration is working well, SCM can determine how to replicate it across many hosts. It aims to control cost and work effort involved in making changes to the software system. The primary goal is to increase productivity with minimal mistakes.

TASKS IN SCM:

1)Configuration Identification-define exam types, categories, authorization validity, maintain security.

2) Change Control-A change request (CR) is submitted and evaluated to assess technical merit, potential side effects, overall impact on other configuration objects and system functions, and the projected cost of the change. The results of the evaluation are presented as a change report, which is used by a change control board (CCB) —a person or group who makes a final decision on the status and priority of the change.

3)Configuration Status- A software

Configuration audit complements the

Formal technical review of the process.

DESIGN DOCUMENT:

TITLE:

Design document for Stock Maintenance System.

INTRODUCTION:

This document gives a symbolic view of the project before the actual implementation.

The main objective of this project is to maintain the details of stocks sold and purchased.All these details are stored in a data base.As soon as the customer purchases a product,the product ID is entered and the sales details are updated.Then,automatically the software updates the profit or loss details of the product,stores the date of purchase,quantity,price,etc.The purpose of this software is to reduce the inefficiency,decrease the dependency on man power and reduce the time for calculation and updation of details. It is user friendly,helps us save our time and increases the efficiency of the process.

LIST OF DIAGRAMS:

1.Use case diagram

2.Class diagram

3.Sequence diagram

4.Collaboration diagram

5.Activity

6.State diagram

7.Component diagram

8.Deployment diagram

DESCRIPTION OF DIAGRAMS:

1.USE CASE DIAGRAM:

Use case diagram is the pictorial representation of how several users interact with the system.It contains actors,functionalities to represented as use case,relationships among use cases.

Use case diagrams are considered for high level requirement analysis of a system. When the requirements of a system are analyzed, the functionalities are captured in use cases.

Use case diagrams specify the events of a system and their flows. But use case diagram never describes how they are implemented. Use case diagram can be imagined as a black box where only the input, output, and the function of the black box is known.



2.CLASS DIAGRAM:

A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

A class notation consists of three parts:

1. **Class Name**
   * The name of the class appears in the first partition.
2. **Class Attributes**
   * Attributes are shown in the second partition.
   * The attribute type is shown after the colon.
   * Attributes map onto member variables (data members) in code.
3. **Class Operations** (Methods)
   * Operations are shown in the third partition. They are services the class provides.
   * The return type of a method is shown after the colon at the end of the method signature.
   * The return type of method parameters are shown after the colon following the parameter name.
   * Operations map onto class methods in code



3.SEQUENCE DIAGRAM:

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function.

We use actors to depict various roles including human users and other external subjects. We represent an actor in a UML diagram using a stick person notation. We can have multiple actors in a sequence diagram.Communication between objects is depicted using messages. The messages appear in a sequential order on the lifeline. We represent messages using arrows.



4.COLLABORATION DIAGRAM:

Collaboration diagrams are used to show how objects interact to perform the behavior of a particular use case, or a part of a use case. Along with sequence diagrams, collaborations are used by designers to define and clarify the roles of the objects that perform a particular flow of events of a use case.

A collaboration diagram resembles aflowchart that portrays the roles, functionality and behavior of individual objects as well as the overall operation of the system in real time. Objects are shown as rectangles with naming labels inside. These labels are preceded by colons and may be underlined. The relationships between the objects are shown as lines connecting the rectangles. Themessages between objects are shown as arrows connecting the relevant rectangles along with labels that define the message sequencing.



5.ACTIVITY:

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another.

The basic purpose of activity diagrams is similar to other four diagrams. It captures the dynamic behavior of the system. Other four diagrams are used to show the message flow from one object to another but activity diagram is used to show message flow from one activity to another.



6.STATE DIAGRAM:

A state diagram is used to represent the condition of the system or part of the system at finite instances of time. It’s a behavioral diagram and it represents the behavior using finite state transitions. State diagrams are also referred to as State machines and State-chart Diagrams. These terms are often used interchangeably. So simply, a state diagram is used to model the dynamic behavior of a class in response to time and changing external stimuli. We can say that each and every class has a state but we don’t model every class using State diagrams.



7.COMPONENT DIAGRAM:

Component diagram is a special kind of diagram in UML. The purpose is also different from all other diagrams discussed so far. It does not describe the functionality of the system but it describes the components used to make those functionalities.

Thus from that point of view, component diagrams are used to visualize the physical components in a system. These components are libraries, packages, files.

Component diagrams can also be described as a static implementation view of a system. Static implementation represents the organization of the components at a particular moment.

A single component diagram cannot represent the entire system but a collection of diagrams is used to represent the whole.



8.DEPLOYMENT DIAGRAM:

Deployment diagram is a structure diagram which shows architecture of the system as deployment (distribution) of software artifacts to deployment targets. Artifacts represent concrete elements in the physical world that are the result of a development process.

A variety of shapes make up deployment diagrams. This list offers an overview of the basic elements you may encounter, and you can see most of these items illustrated in the image below.

* **Artifact:** A product developed by the software, symbolized by a rectangle with the name and the word “artifact” enclosed by double arrows.
* **Association**: A line that indicates a message or other type of communication between nodes.
* **Component:** A rectangle with two tabs that indicates a software element.
* **Dependency:** A dashed line that ends in an arrow, which indicates that one node or component is dependent on another.
* **Interface:** A circle that indicates a contractual relationship. Those objects that realize the interface must complete some sort of obligation.
* **Node:** A hardware or software object, shown by a three-dimensional box.
* **Node as container:** A node that contains another node inside of it.
* **Stereotype:** A device contained within the node, presented at the top of the node, with the name bracketed by double arrows.

