6) Stock Maintenance system:

1. Introduction:

1.1 Purpose of the document

The purpose of this document is to specify the requirements of the stock Maintenance system.

1.2 Scope of the document : This document provides detailed description of the requirements of the stock maintance system. It also includes estimated cost and time development time.

1.3 Overview : It provides efficient platforms to manage inventory in real time, allowing businesses to maintain optimum stock level.

2. General Description:

It is a web-based application to manage stock, providing an efficient way to track inventory levels, handle stock reordering and prevent stockouts.

3. Functional Requirements

- Stock Tracking : Real-time tracking of stock levels across different locations.
- Inventory Update : Automatically update stock levels based on sales, purchases and stock adjustments.
- Stock alerts : Send notifications when stock levels fall below a predefined schedule

4. Interface Requirement:

- User Interface : The system will offer a the web-based interface accessible by administrators and store managers.

- API integration: The system should be able to integrate with existing enterprise resource planning (ERP)

5. Performance Requirements
    - website should be able to handle large number of users.
    - Response time must be short.

6. Design Constraints
    - Designed with html/css, Javascript
    - ~~Use of~~ website must support mysql or mongoDB for database storage.

7. Non-Functional Attributes:
    - Security: ensuring data integrity and confidentiality
    - Relaiability: time to time upgradation.
    - Scalability: scale to support multiple warehouse and store locations.

8. Preliminary Schedule and Budget.
    - Time required: 6 months
    - Estimated Budget: $100,000
        - Requirement analysis -$20,000
        - Development and integration - $30,000
        - Verification and validation - $35,000
        - Evolution and maintainance -$15,000

Stock Maintanance system

WalkInOrder.
- MakePayment()
- CancelOrder()
- updateOrder()

Customer
- customerID: string
- name: String
- # contact: string
- # address: string
- createOrder ()
- cancelOrder ()
- updateDetails()

Order
- ID: string
- totalamount: Double
- date: date
- status: OrderStatus
- placeOrder ()
- cancelOrder()
- confirm Order()
- trackOrder()

1   0..*   Customer Order

OrderStatus
- Confirmed
- Pending
- Delivered
- Cancelled

Supplier
- supplierID: string
- name: String
- contact: Integer[1..*]
- company: String
- supplyProduct()
- generateProduct()
- updateDetails()
- manageOrder()

0..*   1..*

MakesPayment

InitiatePayment

Product
- productID: String
- name: String
- quantity: Integer [0..1]
- price: Double
- updateStock()
- getPrice()
- checkAvailability()
- restock()

Payment
- paymentID: Integer
- price: double
- paymentStatus: String
- Mode: String
- MakePayment()
- status()

Interaction   supplies Product   0..*

1   1

InventoryManagement
- managerID: string
- name: string
- contact: Integer
- email: string
- monitorStock()
- reorderProduct()
- generateReport()
- updateInventory()

managedProduct

0..*