Stock maintainance System

Problem statement
Businesses often struggle with inefficient stock management process, leading to mismatches in inventory levels, delays in stock replenishment. Current systems lack real-time monitoring, user-friendly interfaces and robust reporting capabilities. System that enables seamless stock entry, tracking, movement and reporting.

1. Introduction
1.1 Purpose of this Document
It is to define the specifications and requirements for the development. It serves as a comprehensive guide for the development team, stakeholders and users.

1.2 Scope of this Document
It describes the overall working and objectives of the stock maintainance system. It outlines the value it will provide to users and stakeholders, including efficient management of stock inventory.

1.3 Overview
It is designed to facilitate the management of stock for businesses. It provides functionalities for stock entry, stock tracking and stock movement.

2. General description
The stock entry allows users to add new stock items to the inventory, including details such as item name, quantity, price and supplier information. Provides real-time tracking of stock levels including current stock quantities, locations and availability.
Enables users to track movement of stock items within the organization, including transfers between warehouses, sales and returns.

3. Functional requirements:
3.1 Stock entry
- Ability to add, edit and delete stock items for the inventory database.
- Capture of essential details such as item name, quantity, price, supplier information and other relevant attributes.

3.2 Stock tracking
- Real time monitoring of stock levels, including available quantity, location and status.
- Alerts for low stock levels or stock shortages to facilitate timely replenishment

4. Interface requirements
4.1 User interface
- Intuitive and user friendly interface for easy navigation and data entry.

- Clear display of stock information, including item details, quantities and transaction history.

4.2 System Interfaces
- Integration with barcode scanners for efficient stock entry and tracking.
- Compatibility with external systems for data exchange.

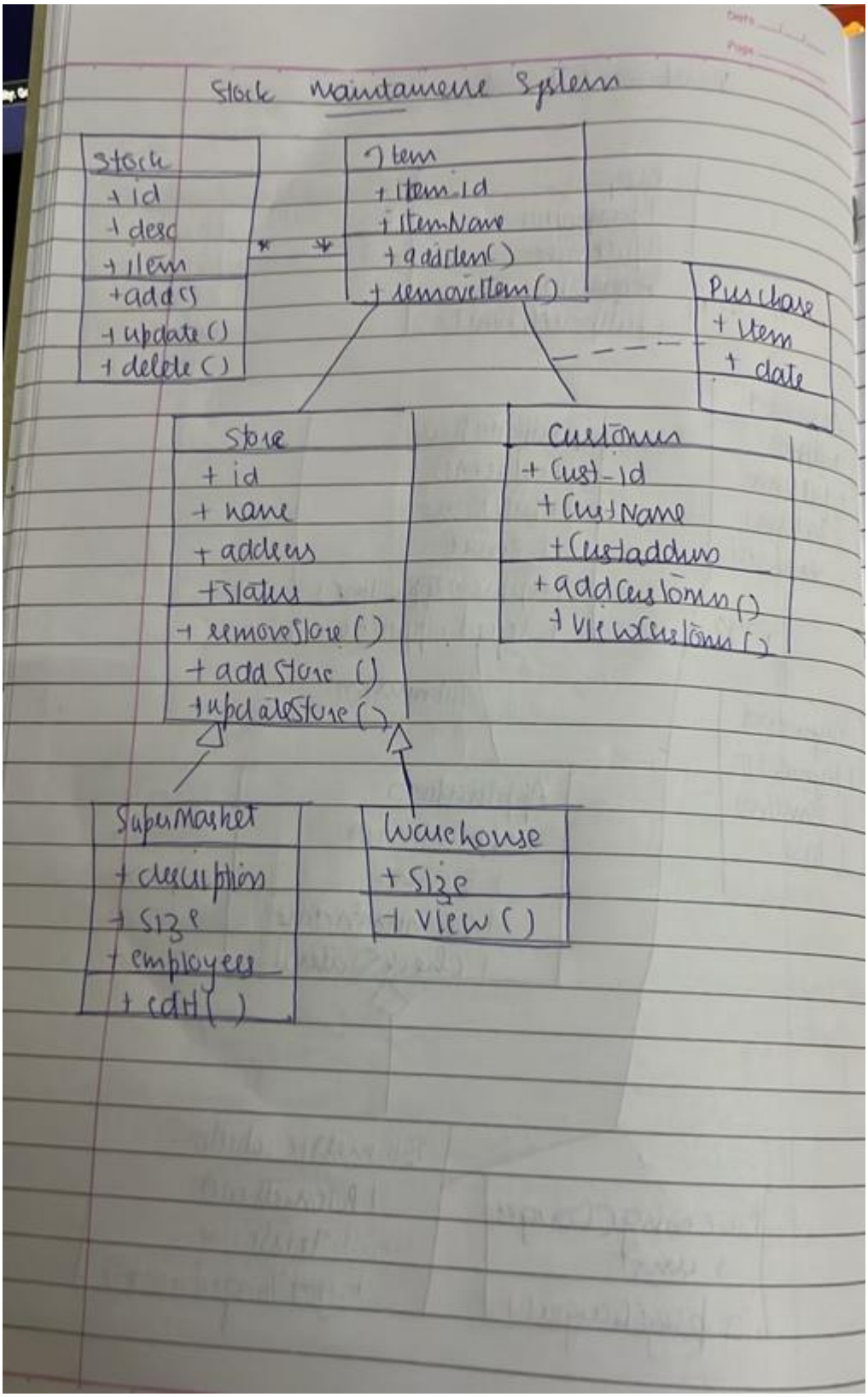
5. Performance Requirements
5.1 Response time.
- Quick response time for stock-related queries and transactions to ensure user productivity.
- Minimal downtime for system maintenance and updates to prevent disruptions in stock management processes.

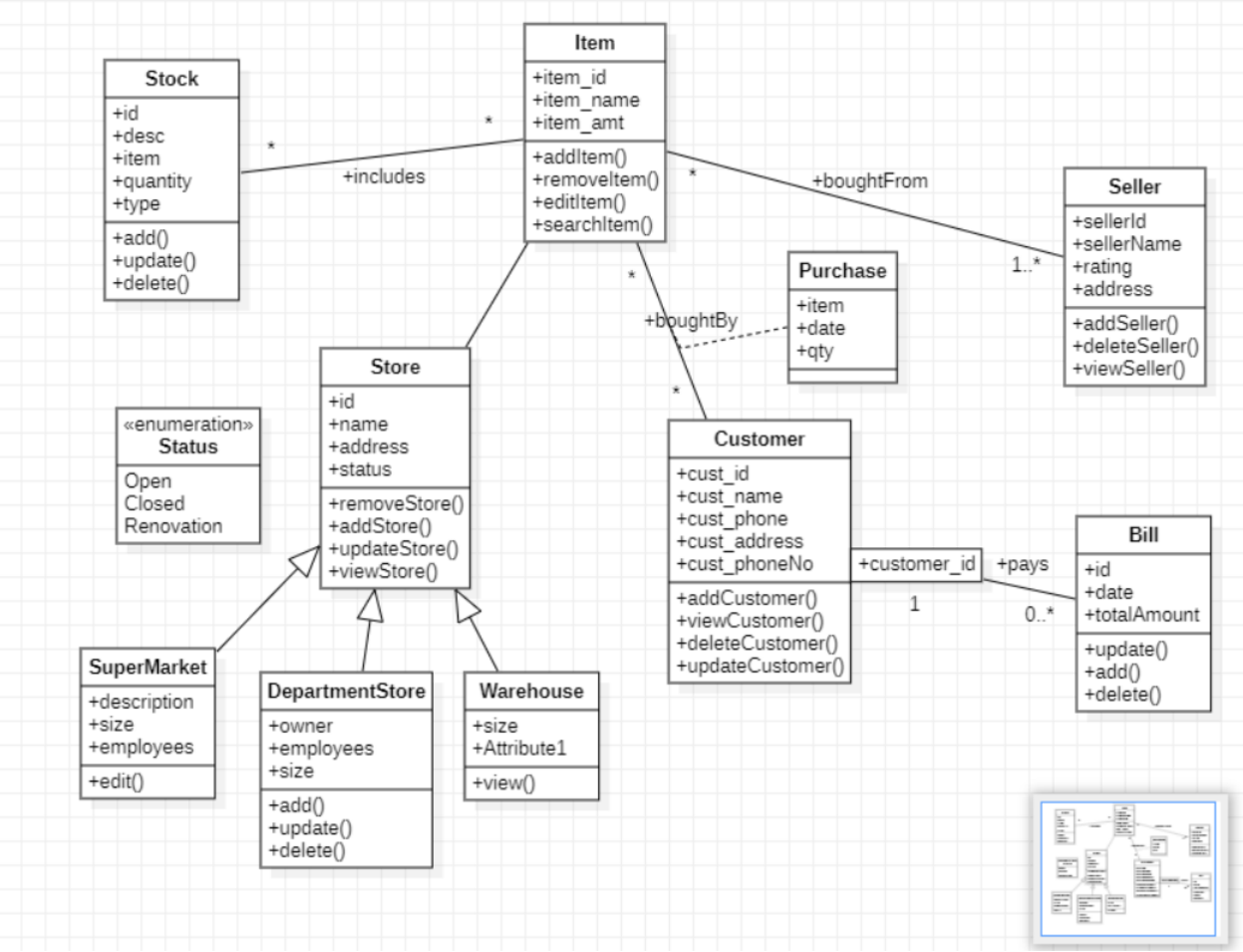
5.2 Scalability
- Ability to handle a large volume of stock items and transactions as the business grows.

6 Design Constraints
6.1 Platform Compatibility
- With various operating systems and web browsers to ensure accessibility.
- Compliance with industry standards for data storage, security and interoperability.

7. Non functional Attributes
- Security - Implementation of user authentication and encryption mechanisms to protect sensitive stock data from unauthorized access.
- Reliable backup and recovery mechanisms to prevent data loss ensure system availability.
- Efficient performance to handle concurrent users and large datasets.

8. Preliminary Schedule and Budget:
- The development is estimated to take approximately 4 months with a budget of $40,000. The schedule includes phases for requirements gathering, design, implementation, testing and deployment


Stock maintanence System
Stock
+ id
+ desc
+ item
+ add()
+ update()
+ delete()
Item
+ item_id
+ itemName
+ addItem()
+ removeItem()
Purchase
+ item
+ date
Store
+ id
+ name
+ address
+ status
+ removeStore()
+ addStore()
+ updateStore()
Customers
+ Cust_id
+ CustName
+ Custaddress
+ addCustomers()
+ ViewCustomers()
SuperMarket
+ description
+ size
+ employees
+ edit()
Warehouse
+ size
+ view()


Stock
+id
+desc
+item
+quantity
+type
+add()
+update()
+delete()
Item
+item_id
+item_name
+item_amt
+addItem()
+removeItem()
+editItem()
+searchItem()
+includes
+boughtFrom
Seller
+sellerId
+sellerName
+rating
+address
+addSeller()
+deleteSeller()
+viewSeller()
Purchase
+item
+date
+qty
+boughtBy
Store
+id
+name
+address
+status
+removeStore()
+addStore()
+updateStore()
+viewStore()
«enumeration»
Status
Open
Closed
Renovation
Customer
+cust_id
+cust_name
+cust_phone
+cust_address
+cust_phoneNo
+addCustomer()
+viewCustomer()
+deleteCustomer()
+updateCustomer()
+customer_id
+pays
Bill
+id
+date
+totalAmount
+update()
+add()
+delete()
SuperMarket
+description
+size
+employees
+edit()
DepartmentStore
+owner
+employees
+size
+add()
+update()
+delete()
Warehouse
+size
+Attribute1
+view()