CURIOSITY
Medicine Shop Automation

SYSTEM ANALYSIS & DESIGN

GROUP 62

Arundhati Banerjee

Meghna Sengupta

CONTENTS

I. SYSTEM ANALYSIS

1. Information Exchange with the Customer

Q. Do the shop's employees need to use the MSA also?

A: No. Presently it is only desired for the shop-owner.

Q: Will a new record be created only when stocks are obtained?

A: Records are created first and updates are made on the already existing record when the medicines are bought.

Q: How many weeks should be considered when calculating threshold value of each medicine?

A: The previous **four weeks** are considered.

2. Background Document

Below is provided the document that has been used to arrive at the Use-case Diagram and the Class Diagram.

The identified actors are shown in italics.

🟥 This colour indicates the use-cases of the actor, 'Shop-owner'.

🟩 This colour indicates the use-cases of the actor, 'System'.

🟦 This colour indicates the use-cases of the secondary actor, 'Printer' .

Medicine Shop Automation (MSA):

Perform structured analysis and structured design for the following Medicine Shop Automation (MSA) software:
A retail medicine shop deals with a large number of medicines procured from various manufacturers. The shop owner maintains different medicines in wall mounted and numbered racks.

- The *shop owner* maintains as few inventory for each item as reasonable, to reduce inventory overheads after being inspired by the **just-in-time (JIT) philosophy**.

- Thus, one important problem the shop owner faces is to be able to order items as soon as the number of items in the inventory reduces below a threshold value. The shop owner wants to maintain medicines to be able to sustain selling for about one week. To calculate the threshold value for each item, the *software* must be able to calculate the average number of medicines sales for one week for each part.
- At the end of each day, the shop owner would request the computer to generate the items to be ordered. The computer should print out the medicine description, the quantity required, and the address of the vendor supplying the medicine. The shop owner should be able to store the name, address, and the code numbers of the medicines that each vendor deals with.
- Whenever new supply arrives, the shop owner would enter the item code number, quantity, batch number, expiry date, and the vendor number. The software should print out a cheque favoring the vendor for the items supplied.
- When the shop owner procures new medicines it had not dealt with earlier, he should be able to enter the details of the medicine such as the medicine trade name, generic name, vendors who can supply this medicine, unit selling and purchasing price. The computer should generate a code number for this medicine which the shop owner would paste the code number in the rack where this medicine would be stored. The shop owner should be able to query about a medicine either using its generic name or the trade name and the software should display its code number and the quantity present.
- At the end of every day the shop owner would give a command to generate the list of medicines which have expired. It should also prepare a vendor-wise list of the expired items so that the shop owner can ask the vendor to replace these items. Currently, this activity alone takes a tremendous amount of labour on the part of the shop owner and is a major motivator for the automation endeavour.
- Whenever any sales occurs, the shop owner would enter the code number of each medicine and the corresponding quantity sold. The MSA should print out the cash receipt.
- The computer should also generate the revenue and profit for any given period. It should also show vendor-wise payments for the period.

3. Drawbacks of the existing System

Accuracy: The existing system is maintained by the shop-owner and is thus, prone to human errors. The new system provides reliability as it largely eliminates the possibility of errors.

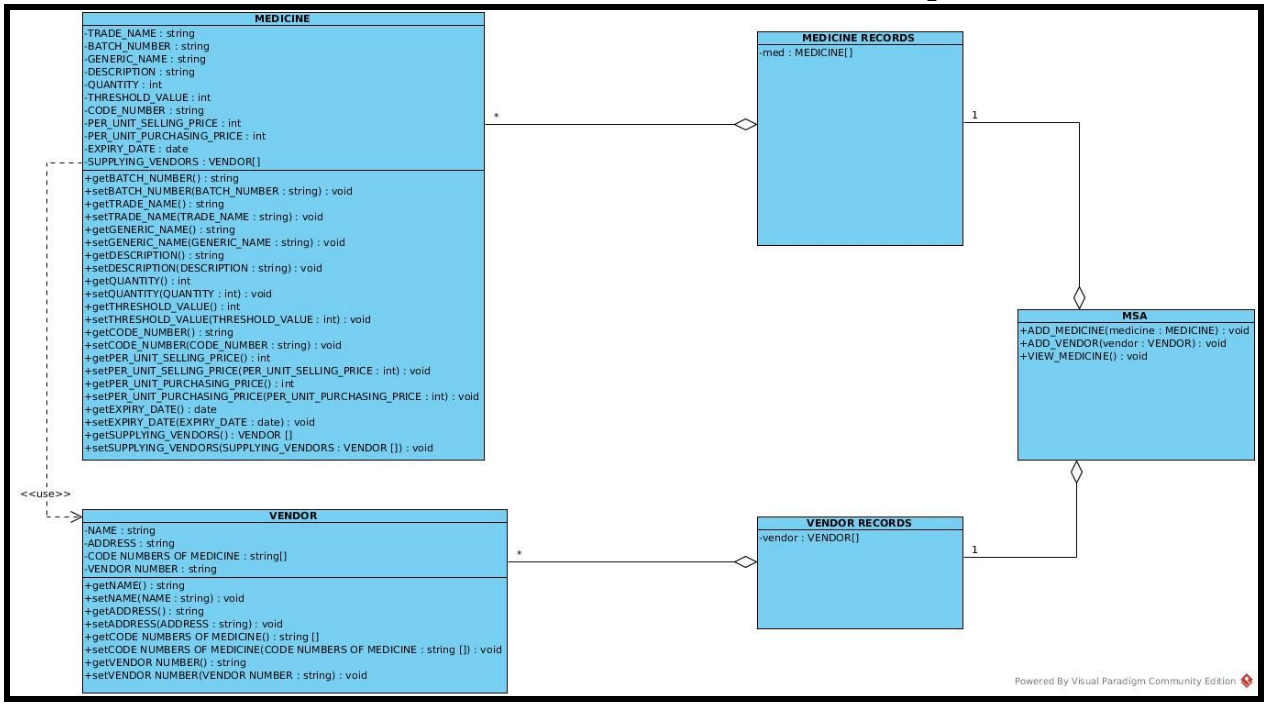
Data Security: Curiosity MSA provides a Login interface which increases the security of the data and reduces the chances of security breach.

Data Integrity: Wider access to well-managed data (using a DBMS) promotes an integrated view of the organization's operations and a clearer view of the big picture.

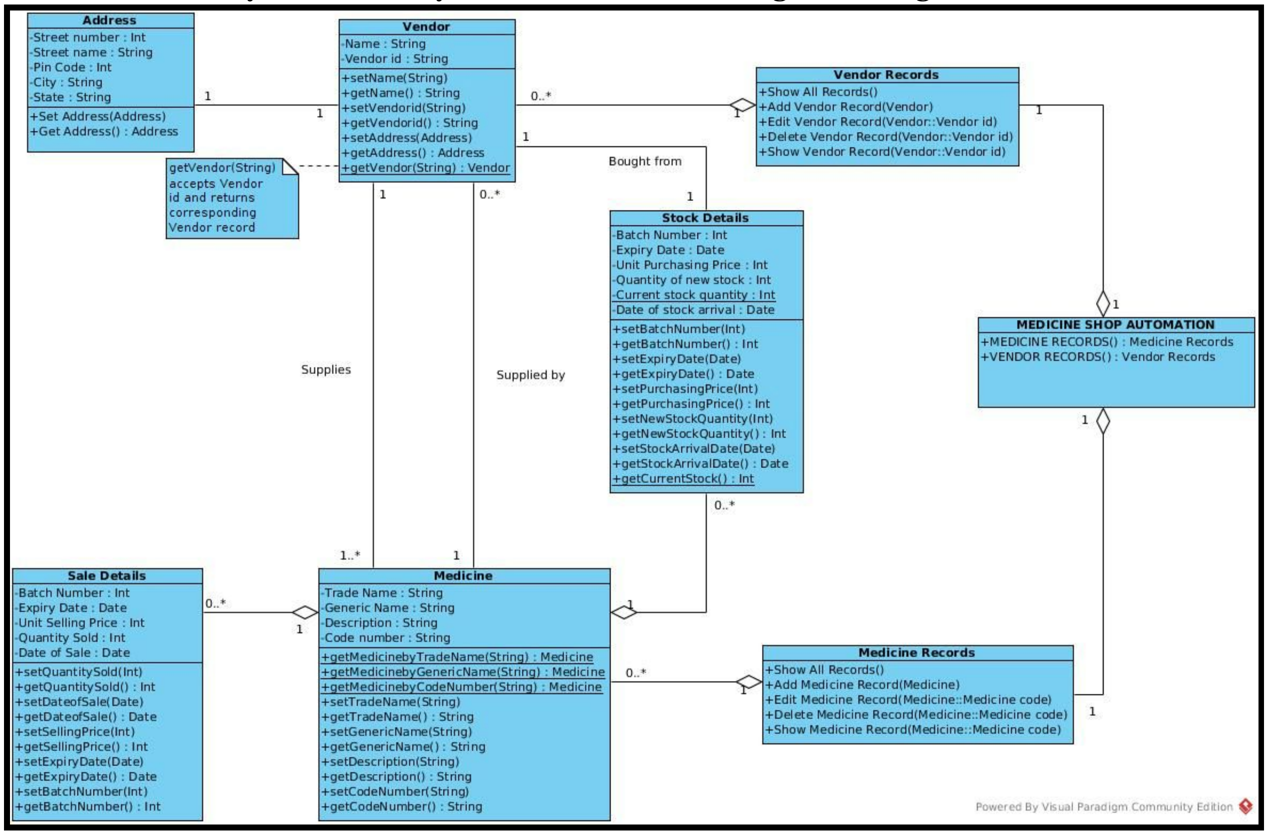
Data Inconsistency: There is a requirement of storing the same information is multiple instances. This results in a chance of data inconsistency whenever some data is altered in a particular instance but not in a separate instance. This problem is eliminated on using a Database Management System as the Curiosity MSA does.

4. Refined Class Diagram

Based on the information available to us, three new classes - “Medicine Records”, “Vendor Records” and “MSA” are added to the class diagram.


MEDICINE
-TRADE_NAME : string
-BATCH_NUMBER : string
-GENERIC_NAME : string
-DESCRIPTION : string
-QUANTITY : int
-THRESHOLD_VALUE : int
-CODE_NUMBER : string
-PER_UNIT_SELLING_PRICE : int
-PER_UNIT_PURCHASING_PRICE : int
-EXPIRY_DATE : date
-SUPPLYING_VENDORS : VENDOR[]
+getBATCH_NUMBER() : string
+setBATCH_NUMBER(BATCH_NUMBER : string) : void
+getTRADE_NAME() : string
+setTRADE_NAME(TRADE_NAME : string) : void
+getGENERIC_NAME() : string
+setGENERIC_NAME(GENERIC_NAME : string) : void
+getDESCRIPTION() : string
+setDESCRIPTION(DESCRIPTION : string) : void
+getQUANTITY() : int
+setQUANTITY(QUANTITY : int) : void
+getTHRESHOLD_VALUE() : int
+setTHRESHOLD_VALUE(THRESHOLD_VALUE : int) : void
+getCODE_NUMBER() : string
+setCODE_NUMBER(CODE_NUMBER : string) : void
+getPER_UNIT_SELLING_PRICE() : int
+setPER_UNIT_SELLING_PRICE(PER_UNIT_SELLING_PRICE : int) : void
+getPER_UNIT_PURCHASING_PRICE() : int
+setPER_UNIT_PURCHASING_PRICE(PER_UNIT_PURCHASING_PRICE : int) : void
+getEXPIRY_DATE() : date
+setEXPIRY_DATE(EXPIRY_DATE : date) : void
+getSUPPLYING_VENDORS() : VENDOR []
+setSUPPLYING_VENDORS(SUPPLYING_VENDORS : VENDOR []) : void
*
MEDICINE RECORDS
-med : MEDICINE[]
1
MSA
+ADD_MEDICINE(medicine : MEDICINE) : void
+ADD_VENDOR(vendor : VENDOR) : void
+VIEW_MEDICINE() : void
<<use>>
VENDOR
-NAME : string
-ADDRESS : string
-CODE NUMBERS OF MEDICINE : string[]
-VENDOR NUMBER : string
+getNAME() : string
+setNAME(NAME : string) : void
+getADDRESS() : string
+setADDRESS(ADDRESS : string) : void
+getCODE NUMBERS OF MEDICINE() : string []
+setCODE NUMBERS OF MEDICINE(CODE NUMBERS OF MEDICINE : string []) : void
+getVENDOR NUMBER() : string
+setVENDOR NUMBER(VENDOR NUMBER : string) : void
*
VENDOR RECORDS
-vendor : VENDOR[]
1
Powered By Visual Paradigm Community Edition

After further analysis, we finally arrived at the following class diagram :



Here, the Class Diagram for the MSA has 2 classes - the Vendor Records and the Medicine Records, the former having 0 or more records of the class Vendor and the latter for class Medicine.

Each Vendor has an address, which is indicated by the **association** between Vendor class and Address class. The purpose of creating Address as a separate class is that, address generally has a fixed format that needs to be followed. Moreover, it can be utilised for **code reuse** when other classes are introduced, like the Customer or Employee - an instance of each of which will have an address.

Moreover, each Vendor supplies one or more Medicines, hence there is an **aggregation** relation between the classes Vendor and Medicine (each instance of Vendor has an aggregation of 1 or more instances for Medicine) as shown in the Class Diagram above.

The class Medicine is structured to contain as its attributes the bare minimum details that can describe a medicine. Further, Medicine has an **aggregation** relationship with both class Sale Details and Stock Details. So, each instance of Medicine, i.e. every record of medicine stored in the database will have an aggregation of 0 or more instances of Sale Details. An instance of Sale Details is created every time that medicine is sold. Besides, each instance of Medicine will also have an aggregation of 0 or more instances of Stock Details. An instance of Stock Details is created every time new stock for that medicine arrives.

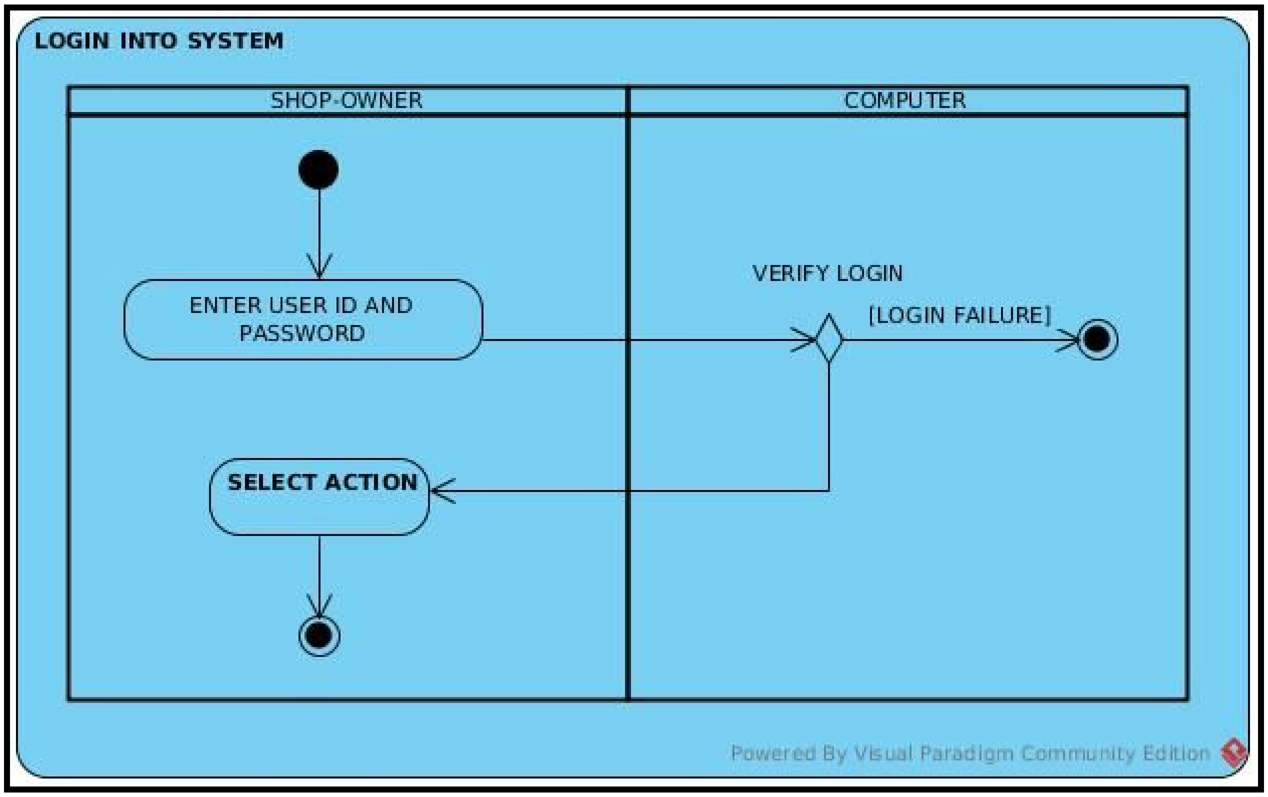
In addition, Medicine class also has a “supplied by” **association** to Vendor class. This is because ach record of Medicine must be accompanied by the corresponding records (1 or more) of Vendors who supply that medicine to the shop.

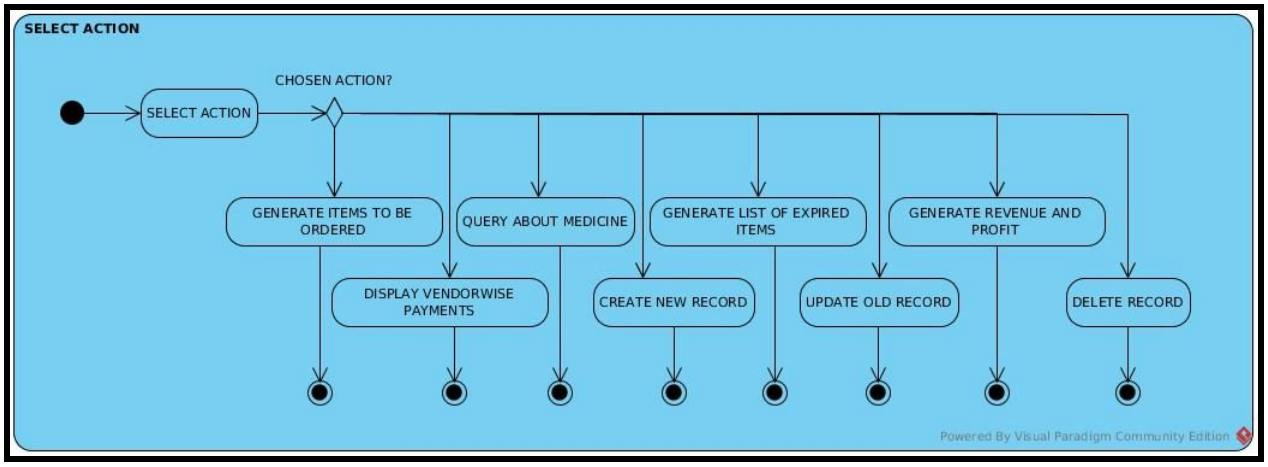
There is also a “bought from” **association** between Stock Details and Vendor since with each stock arrival, the vendor who supplied the medicines must also be recorded.

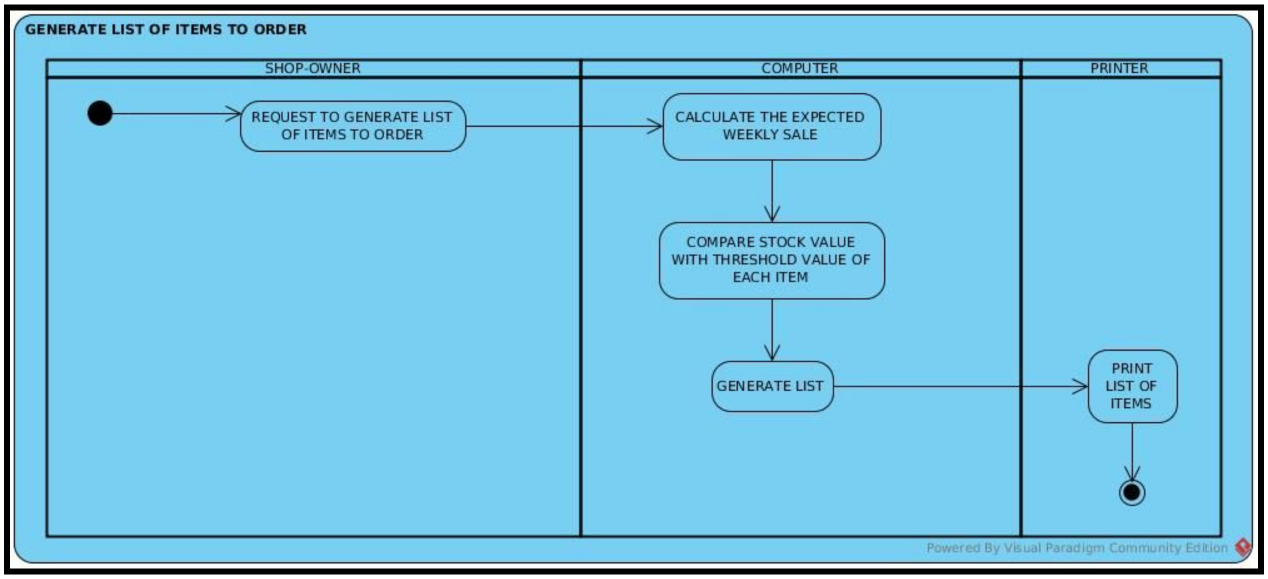
The attributes and operations for each class are as shown in the diagram. A few class operations have been introduced for easier access of records.

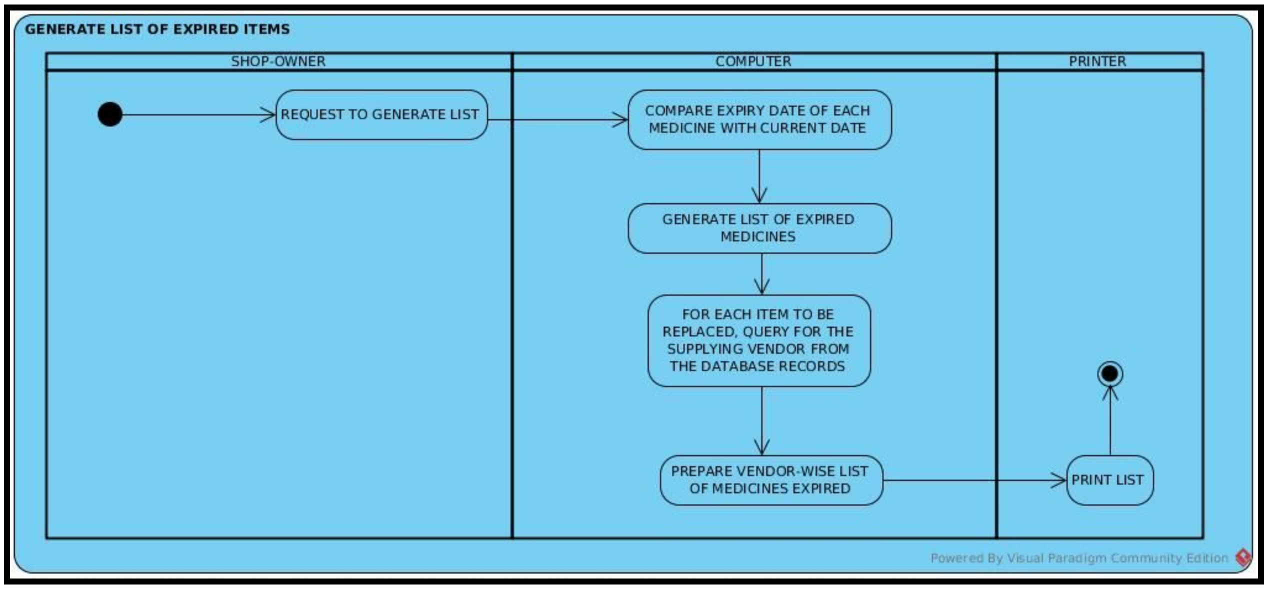
5. Activity Diagram

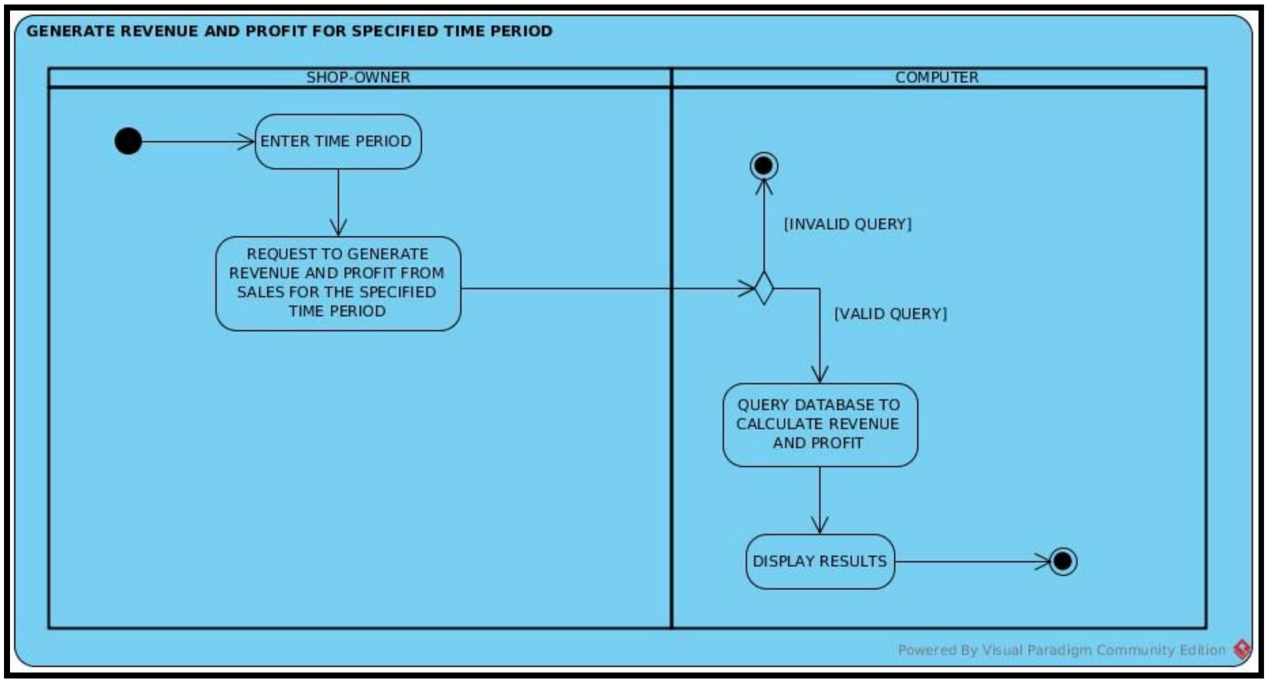
The following are the activity diagrams for the MSA:

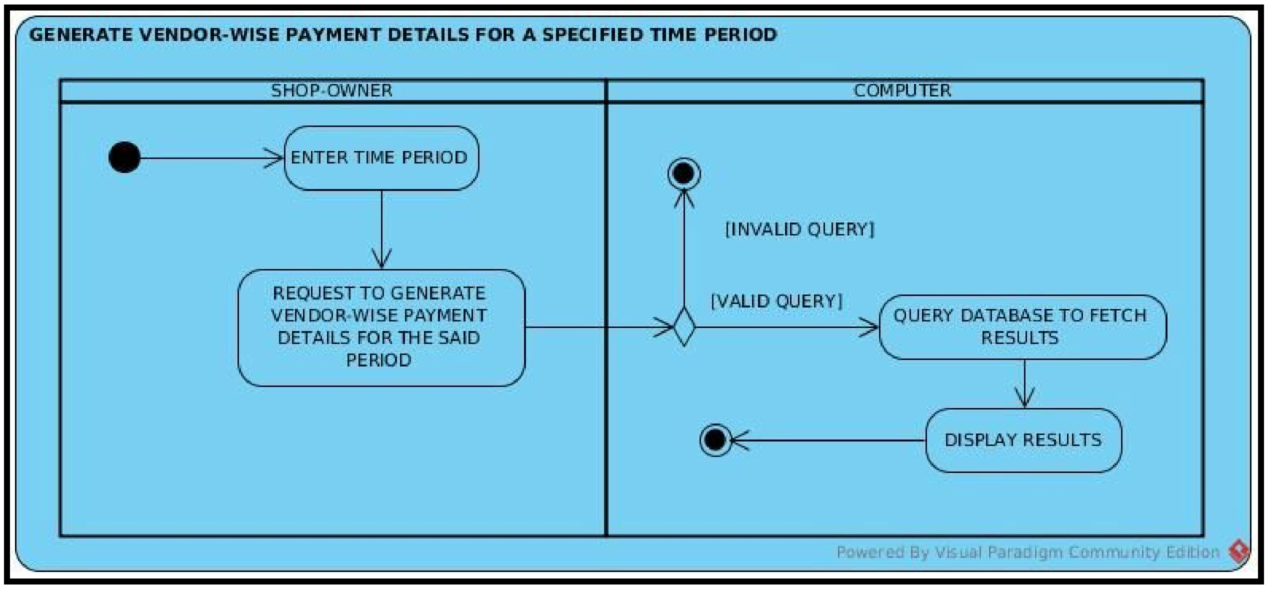

LOGIN INTO SYSTEM
SHOP-OWNER
COMPUTER
ENTER USER ID AND PASSWORD
VERIFY LOGIN
[LOGIN FAILURE]
SELECT ACTION
Powered By Visual Paradigm Community Edition

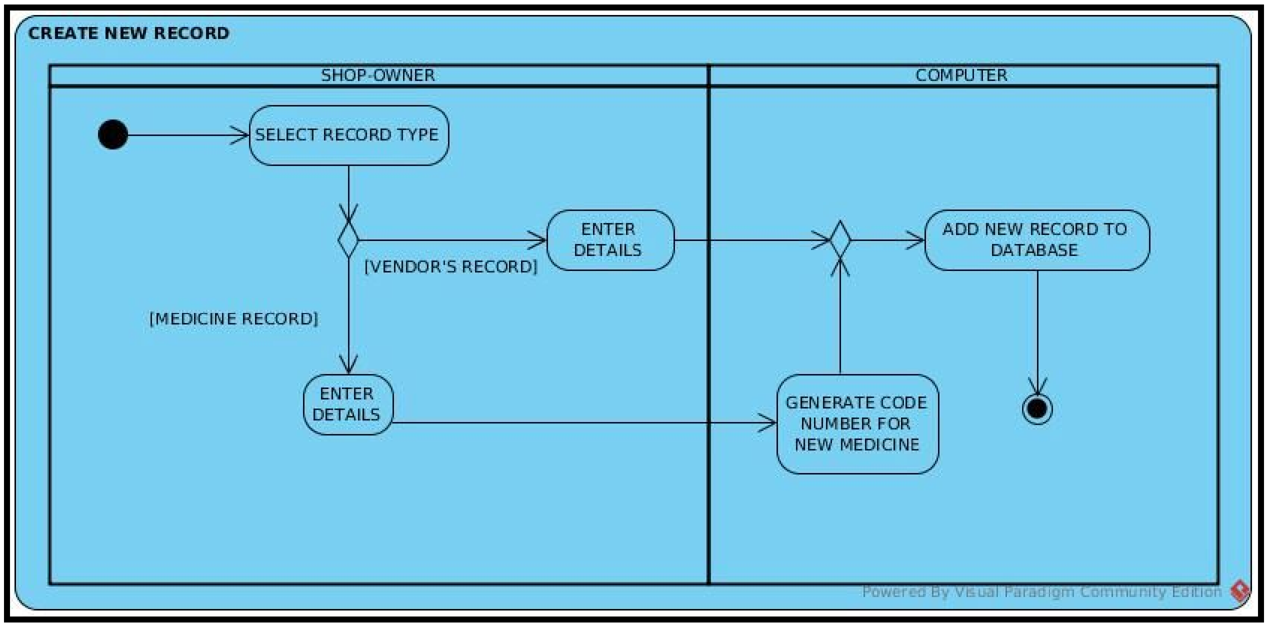

SELECT ACTION
SELECT ACTION
CHOSEN ACTION?
GENERATE ITEMS TO BE ORDERED
QUERY ABOUT MEDICINE
GENERATE LIST OF EXPIRED ITEMS
GENERATE REVENUE AND PROFIT
DISPLAY VENDORWISE PAYMENTS
CREATE NEW RECORD
UPDATE OLD RECORD
DELETE RECORD
Powered By Visual Paradigm Community Edition

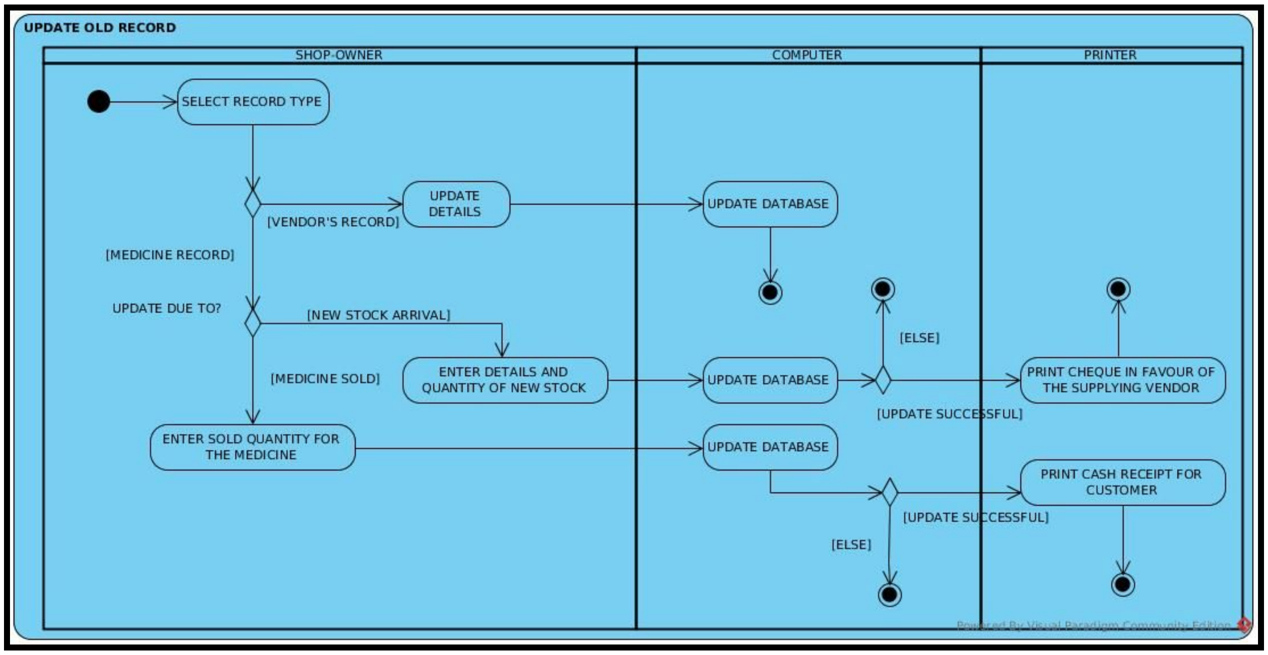

GENERATE LIST OF ITEMS TO ORDER
SHOP-OWNER
COMPUTER
PRINTER
REQUEST TO GENERATE LIST OF ITEMS TO ORDER
CALCULATE THE EXPECTED WEEKLY SALE
COMPARE STOCK VALUE WITH THRESHOLD VALUE OF EACH ITEM
GENERATE LIST
PRINT LIST OF ITEMS
Powered By Visual Paradigm Community Edition

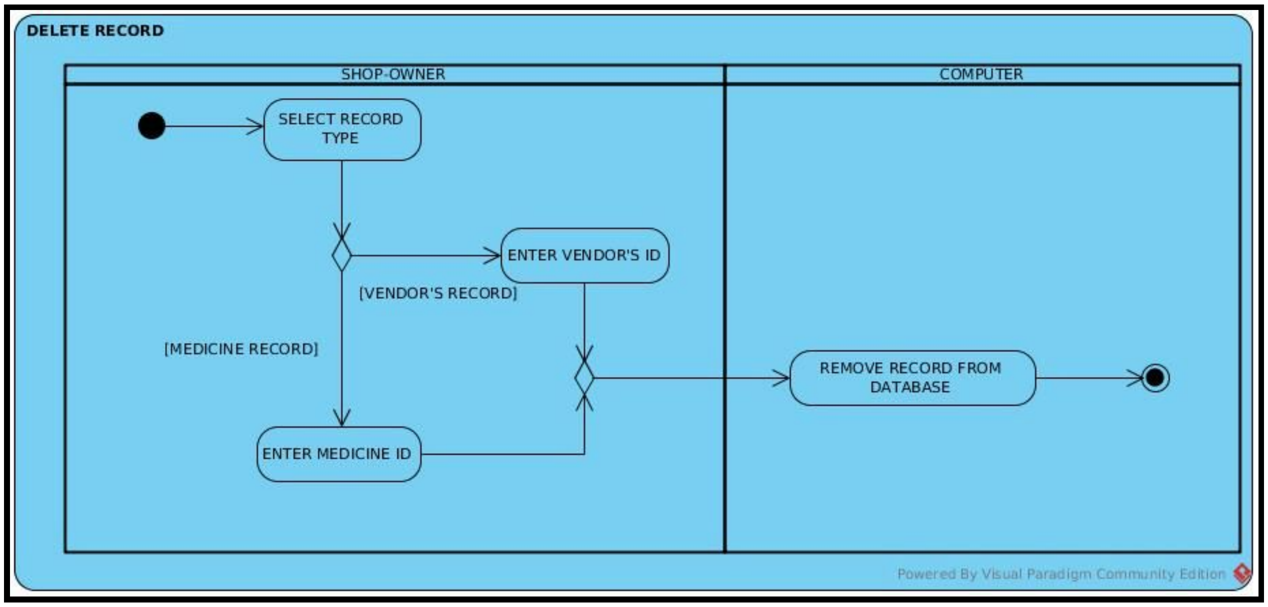

GENERATE LIST OF EXPIRED ITEMS
SHOP-OWNER
COMPUTER
PRINTER
REQUEST TO GENERATE LIST
COMPARE EXPIRY DATE OF EACH MEDICINE WITH CURRENT DATE
GENERATE LIST OF EXPIRED MEDICINES
FOR EACH ITEM TO BE REPLACED, QUERY FOR THE SUPPLYING VENDOR FROM THE DATABASE RECORDS
PREPARE VENDOR-WISE LIST OF MEDICINES EXPIRED
PRINT LIST
Powered By Visual Paradigm Community Edition


GENERATE REVENUE AND PROFIT FOR SPECIFIED TIME PERIOD
SHOP-OWNER
COMPUTER
ENTER TIME PERIOD
REQUEST TO GENERATE REVENUE AND PROFIT FROM SALES FOR THE SPECIFIED TIME PERIOD
[INVALID QUERY]
[VALID QUERY]
QUERY DATABASE TO CALCULATE REVENUE AND PROFIT
DISPLAY RESULTS
Powered By Visual Paradigm Community Edition


GENERATE VENDOR-WISE PAYMENT DETAILS FOR A SPECIFIED TIME PERIOD
SHOP-OWNER
COMPUTER
ENTER TIME PERIOD
REQUEST TO GENERATE VENDOR-WISE PAYMENT DETAILS FOR THE SAID PERIOD
[INVALID QUERY]
[VALID QUERY]
QUERY DATABASE TO FETCH RESULTS
DISPLAY RESULTS
Powered By Visual Paradigm Community Edition


CREATE NEW RECORD
SHOP-OWNER
COMPUTER
SELECT RECORD TYPE
[VENDOR'S RECORD]
ENTER DETAILS
[MEDICINE RECORD]
ENTER DETAILS
ADD NEW RECORD TO DATABASE
GENERATE CODE NUMBER FOR NEW MEDICINE
Powered By Visual Paradigm Community Edition


UPDATE OLD RECORD
SHOP-OWNER
COMPUTER
PRINTER
SELECT RECORD TYPE
[VENDOR'S RECORD]
UPDATE DETAILS
UPDATE DATABASE
[MEDICINE RECORD]
UPDATE DUE TO?
[NEW STOCK ARRIVAL]
[MEDICINE SOLD]
ENTER DETAILS AND QUANTITY OF NEW STOCK
UPDATE DATABASE
[ELSE]
[UPDATE SUCCESSFUL]
PRINT CHEQUE IN FAVOUR OF THE SUPPLYING VENDOR
ENTER SOLD QUANTITY FOR THE MEDICINE
UPDATE DATABASE
[UPDATE SUCCESSFUL]
[ELSE]
PRINT CASH RECEIPT FOR CUSTOMER
Powered By Visual Paradigm Community Edition



6. Specifications

6.1. System Parameters

Processor: Pentium 4

RAM: 512 MB RAM

Monitor: 15” Colour Monitor

Processor Speed: 1.7 GHz

6.2. Platform

The System is designed to run on both Windows and Linux platforms. The Curiosity MSA provides a lot of flexibility regarding the platform of choice.

6.3. Language

The automation is primarily coded using JAVA. This language was chosen because of the numerous benefits it provides for GUI-based design. GUI in JAVA is greatly user-friendly and provides much more scope for interaction between the user and the system. Also, in case of updates in the MSA, JAVA can be very useful and easy to use.

6.4. Libraries

The Front end of the MSA has been developed using JAVA. The Libraries used are mainly the basic libraries like the standard Input/Output Library, the utilities library and the Text library. In addition, for the design of the GUI, a number of Abstract Window Toolkits (AWTs) are utilised like the JFrame awt.

The Back end however, has been developed using MySQL which helps in the secure storage of all medicine and vendor records for the medicine shop.

6.5. Database Management

The database of the medicine shop is managed using the MySQL software. This makes the MSA easy-to-use and reliable. It also provides the required security for the confidential and significant records of the medicine shop.

Additionally, MySQL is fast and runs on several different operating systems, thus, preserving the supportability of the Curiosity MSA.

7. Performance and Limitations

The Curiosity MSA is an easy to use, quick to deploy medicine shop automation system that increases visibility and control of all pharmacy management processes.

The Shop-owner benefits from increased accuracy and improved service levels. In addition, the MSA provides a wide range of functionality features, such as responding to various user-specific queries, printing out cheques and shop-receipts, etc.

The Curiosity MSA is also a very cost-effective solution - helping to reduce the inventory management costs.

LIMITATIONS:

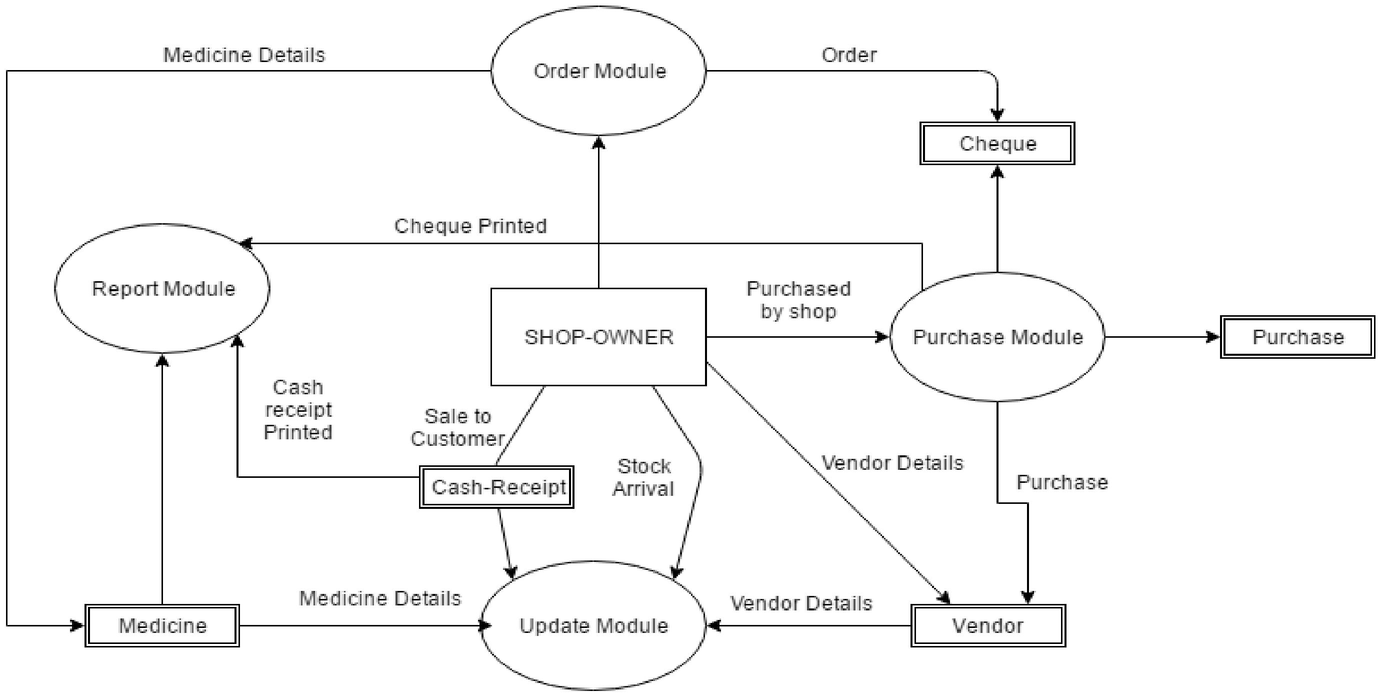
The Curiosity MSA, as of yet, does not provide any kind of access to the other employees of the shop apart from the shop owner. Both the medicine and the vendor records are completely inaccessible to the other employees of the shop. While this increases information security, on the other hand, it can at times, become very difficult for the owner alone to manage.

8. System Test Plan

Use Case	Function being tested	Initial System State	Input	Output
System Startup	System boots properly	System is off	Press 'ON' button	System requests username & password
Login	Accepts correct password and rejects incorrect ones	System is on	Enter username and password	Opens interface if username & password is correct, otherwise shows error
Creation of record	New record can be created	Options are displayed	Click the 'create' button and enter information	Displays a message if information is correct, otherwise displays error message and returns to option screen
Updating Record	Existing record can be updated	Options are displayed	Click the 'update' button and enter information	Displays a message if information is correct, otherwise displays error message and returns to option screen
Search	Returns accurate details on searching by appropriate parameters	Options are displayed	Click the appropriate 'search' button and enter information	Displays required details if information is correct, otherwise displays error message and returns to option screen
Queries	Returns proper solutions to various user-specific queries	Options are displayed	Click the appropriate query button	Returns required solution
Deleting Record	Existing record can be smoothly removed	Interface is displayed	Click on the 'delete' button	Shows a message that required record has been deleted

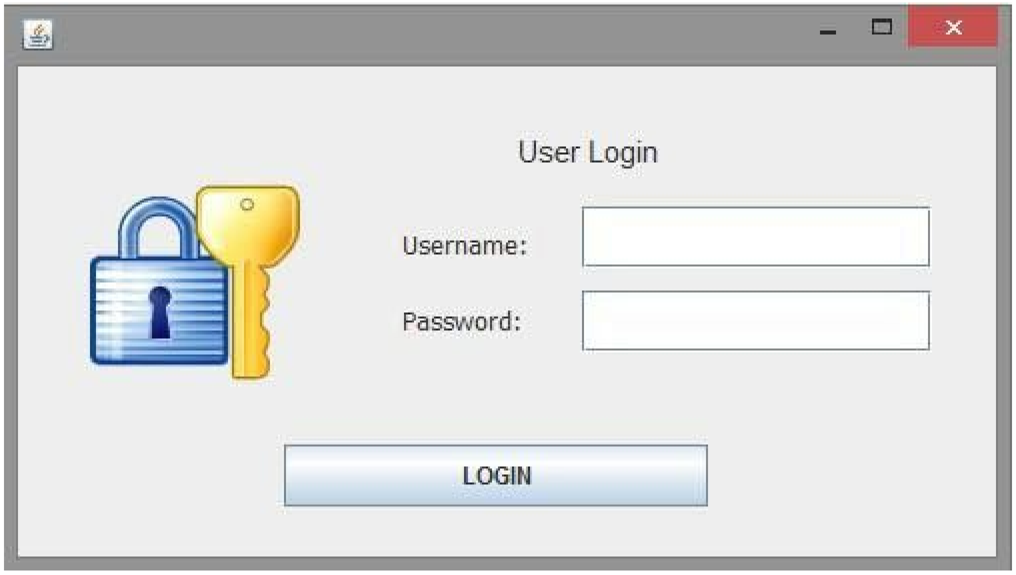
II. SYSTEM DESIGN

1. Data-Flow Diagram

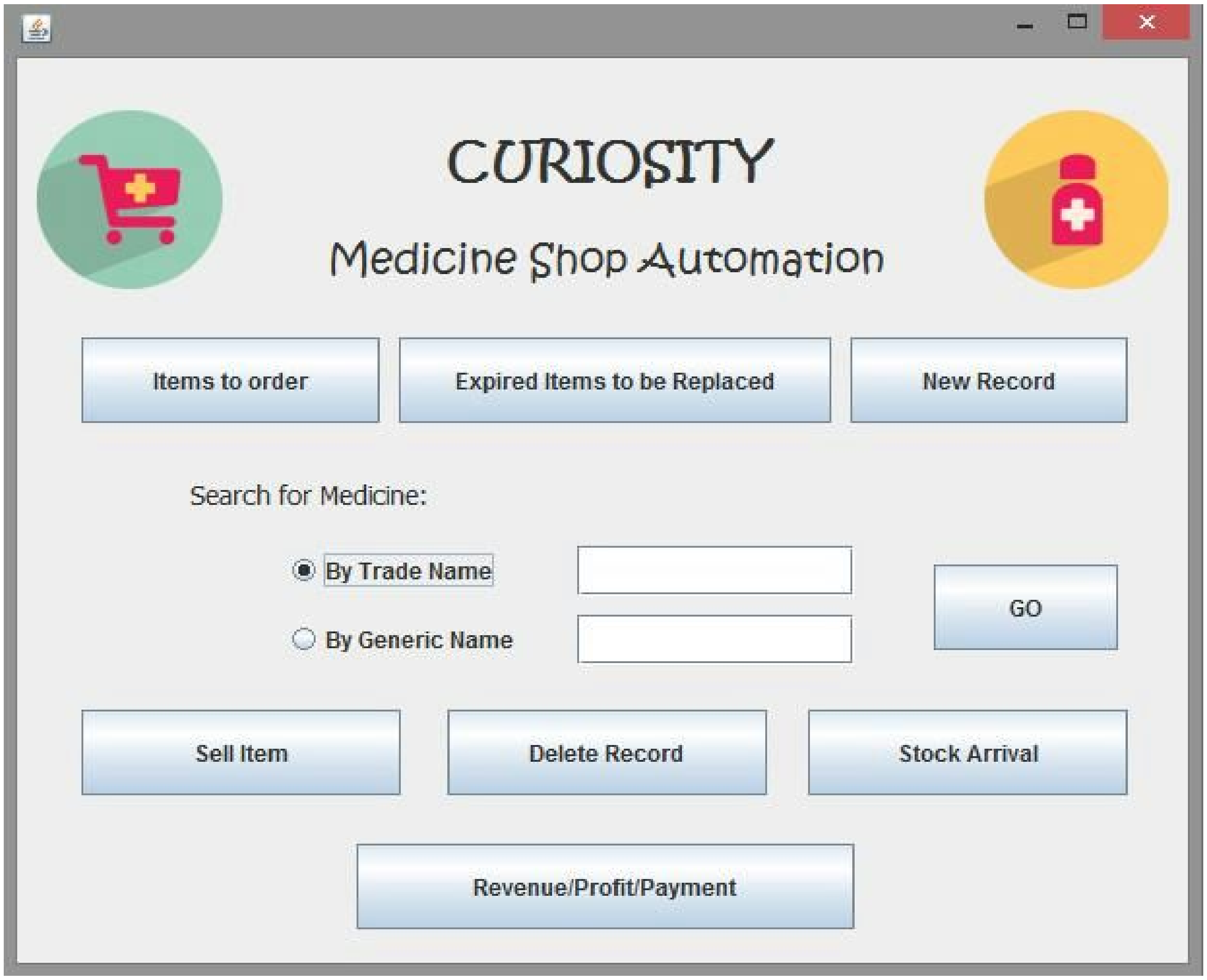


2. Interface in Target Language

A. LOGIN PAGE



B. WELCOME PAGE



3. Exception Design

<u>EXCEPTIONS:</u>

- <u>Date of Expiry out of Range</u> -

This Exception case arises when the Date of Expiry of any particular batch of medicine that has been purchased by the shop is found to have an Expiry date that is before a certain date (pre-specified by the owner).

- <u>Insufficient stocks Received</u> -

This exception is thrown when even the purchased amount of medicines does not raise the quantity of medicines available in the shop to more than the threshold value

- <u>Wrong information Entered</u> -

Whenever information is entered in the wrong format (Eg. when any char value is entered for a field that is supposed to contain, say, a long value), this exception is thrown.

When this Exception is thrown, the user is asked to repeat the process until he/she enters some valid information.

4. Reuse Identified

Presently, the Curiosity MSA does not implement any reuse. However, in future versions of the MSA, if the use of the MSA is further extended to provide service to the customers and shop-employees as well, there is a scope for code reuse. The class “Address” will then be reused in the classes “Vendor”, “Employee” and “Customer”.