MEDICINE SHOP AUTOMATION

Test Suite Design Document & Results

Group 62 –

Meghna Sengupta

Arundhati Banerjee

CONTENTS

- INTRODUCTION
- TEST OBJECTIVE
- OVERALL TEST PLAN
- TEST CASES AND RESULTS
  - UNIT TESTING -
    - BLACK BOX TESTS
    - WHITE BOX TESTS
  - SYSTEM TESTING -
    - FUNCTION VALIDATION TESTING
    - PERFORMANCE TESTING

INTRODUCTION

This document is an overview defining testing strategy for the Medicine Shop Automation software.Its objective is to communicate project-wide quality standards and procedures. It portrays a snapshot of the project as of the end of the planning phase. This document will address the different standards that will apply to the unit, integration and system testing of the specified application. Testing criteria under the white box, black box, and system-testing paradigm have been specified.

TEST OBJECTIVE

The objective of this test plan is to find and report as many bugs as possible to improve the integrity of our software. Although exhaustive testing is not possible, a broad range of tests have been exercised to achieve the goal. The primary aim is to verify whether all the functionalities specified in the SRS document have been tested and to check for the correctness of their implementation.

OVERALL 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

TEST CASES AND RESULTS

❑ UNIT TESTING :-

➢ SYSTEM STARTUP :

BLACK BOX TEST: The system boots up properly at a prompt from the shop-owner.

Expected system behaviour: Login Page should open prompting a username and password from the user.

Output:

➢ LOGIN:

BLACK BOX TEST : The shop-owner can login into the system by providing the set username and password.

Parameters : Username, Password

System-acceptable values of the parameters:

Username : MSA

Password : 12345

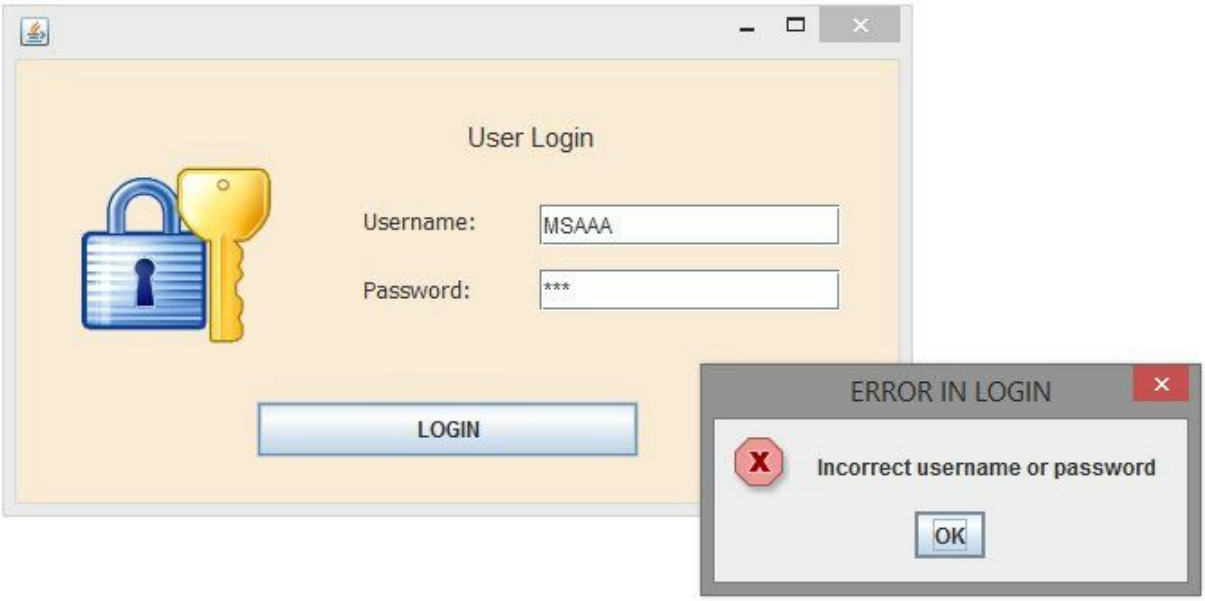
Expected system behaviour :

Correct username and password : The shop-owner is logged in with a success message and presented with the welcome page to access other functionalities.

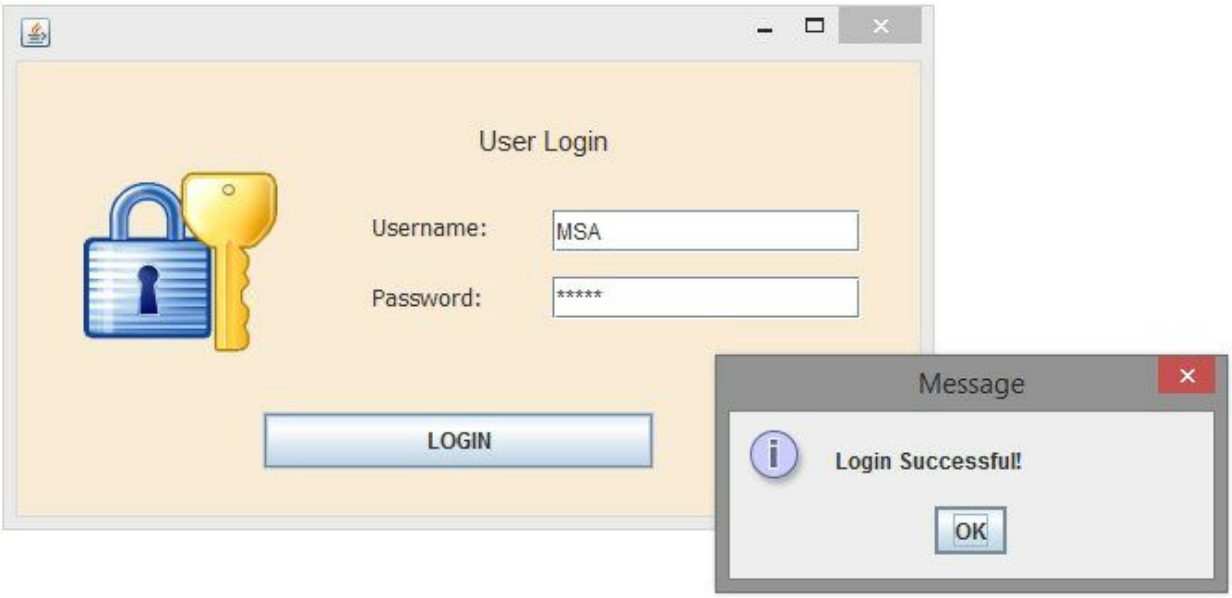
Invalid username and/or password : An error message pops-up informing the user.

Test cases:

- INVALID USERNAME AND/OR PASSWORD:



➢ VALID USERNAME AND PASSWORD :



➢CREATION OF RECORD:

○Creating a medicine record:

BLACK BOX TEST : The shop-owner can enter the record for a new medicine that has been procured by the shop.

Parameters : Trade Name, Generic Name, Description, Quantity, Threshold Value

System-acceptable values of the parameters:

Trade Name: Any string of less than 45 characters.

Generic Name: Any string of less than 45 characters.

Description: Any string of less than 150 characters.

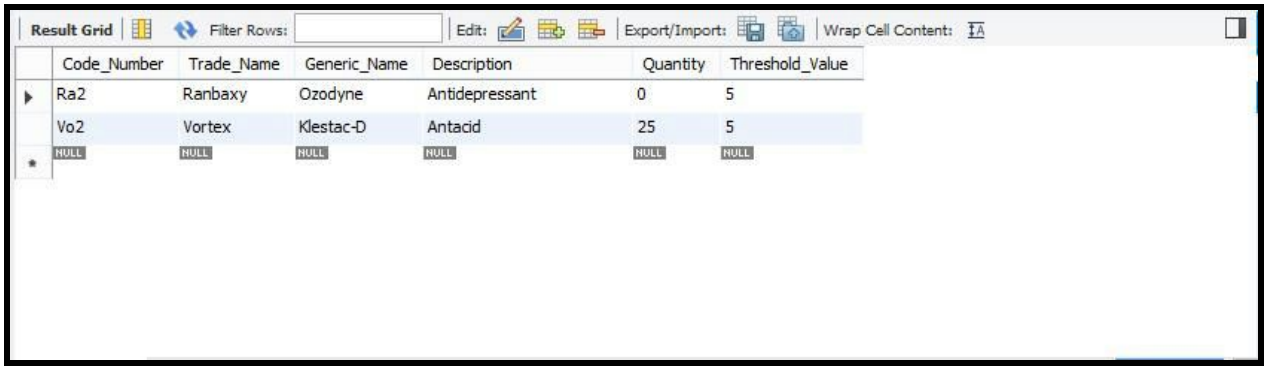
Quantity: Any integer

Threshold Value: Any integer

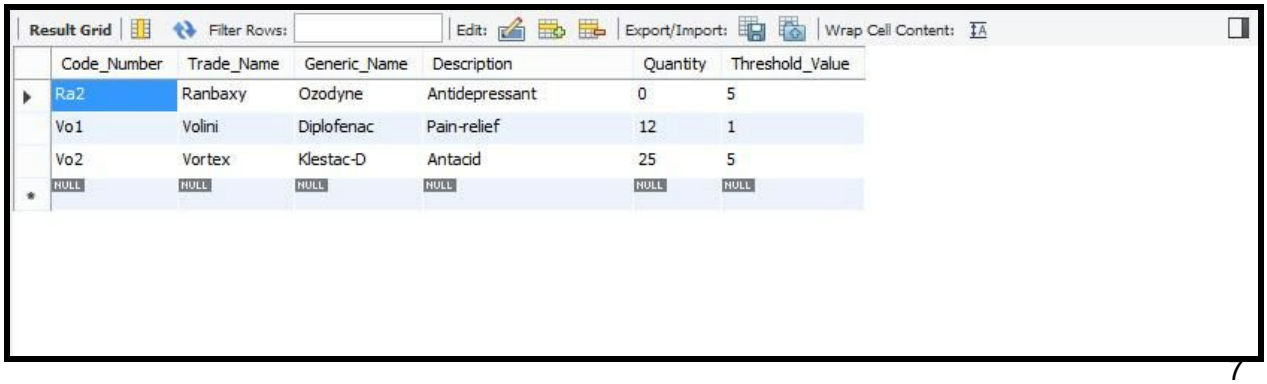
Expected system behaviour :

Acceptable data types : The system-generated code number for the medicine is displayed. The shop-owner is informed via a message dialog box that the new record has been created.

<u>WHITE BOX TESTING</u> : Checking that the database is correspondingly updated in the backend.



Code_Number	Trade_Name	Generic_Name	Description	Quantity	Threshold_Value
Ra2	Ranbaxy	Ozodyne	Antidepressant	0	5
Vo2	Vortex	Klestac-D	Antacid	25	5
NULL	NULL	NULL	NULL	NULL	NULL



Code_Number	Trade_Name	Generic_Name	Description	Quantity	Threshold_Value
Ra2	Ranbaxy	Ozodyne	Antidepressant	0	5
Vo1	Volini	Diplofenac	Pain-relief	12	1
Vo2	Vortex	Klestac-D	Antacid	25	5
NULL	NULL	NULL	NULL	NULL	NULL

Incorrect data types : An error message pops-up informing the user.

○Creating a vendor record:

BLACK BOX TEST : The shop-owner can enter the record for a new vendor who the shop engages in trade with.

Parameters : Name, Address.

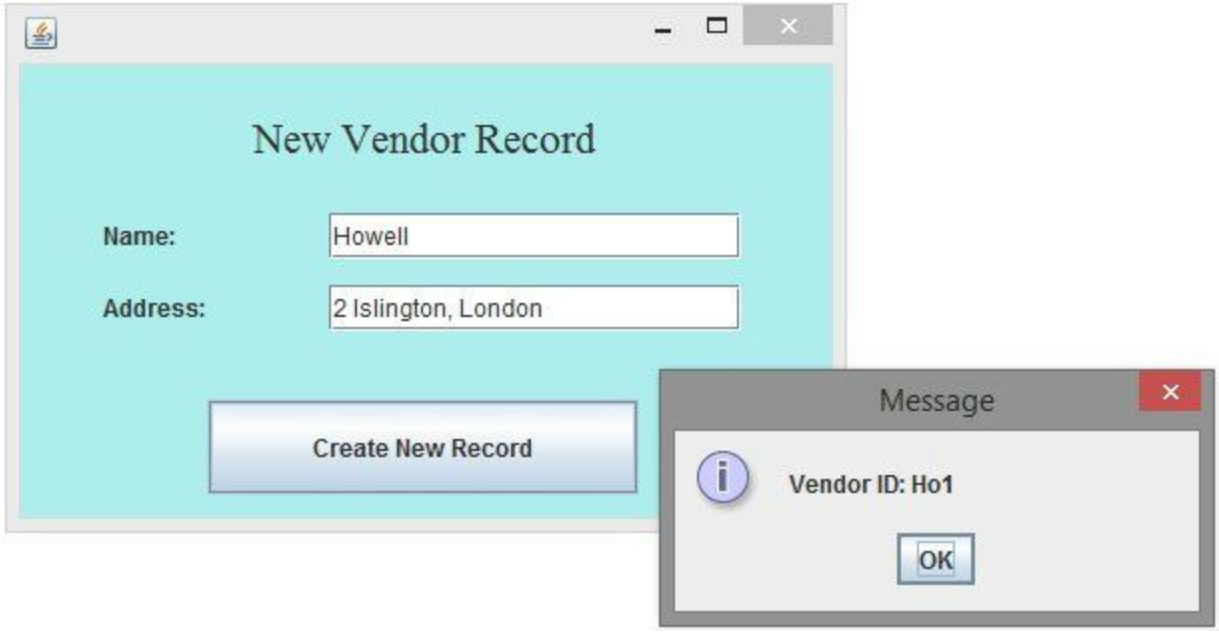
System-acceptable values of the parameters:

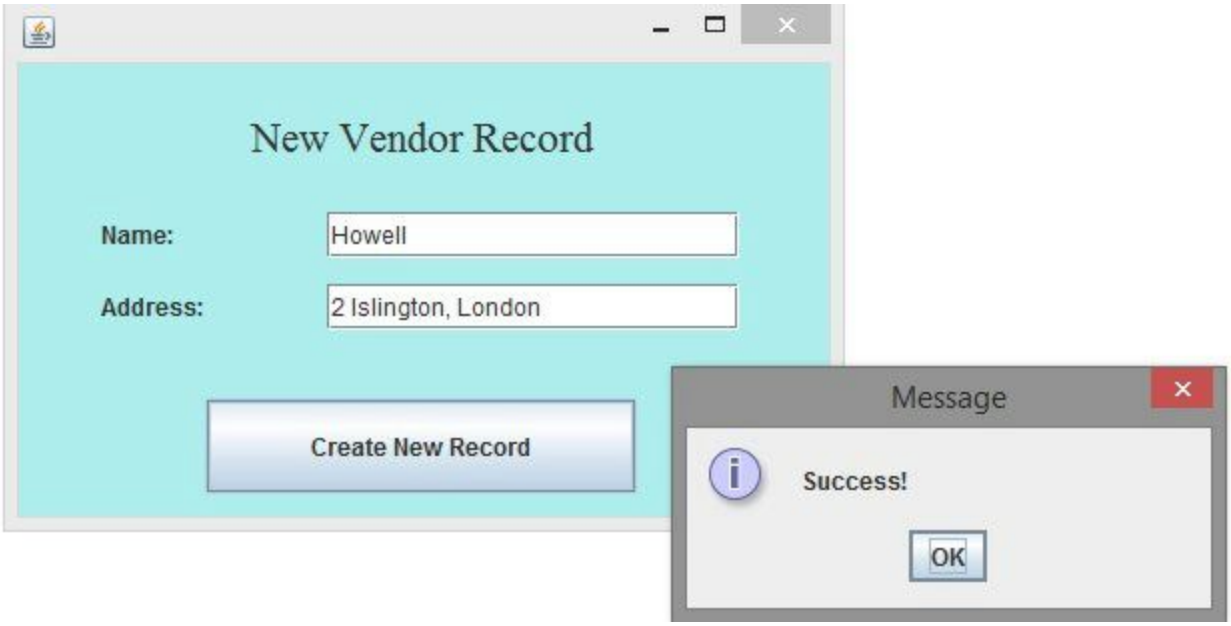
Name: Any string of less than 45 characters.

Address: Any string of less than 150 characters.

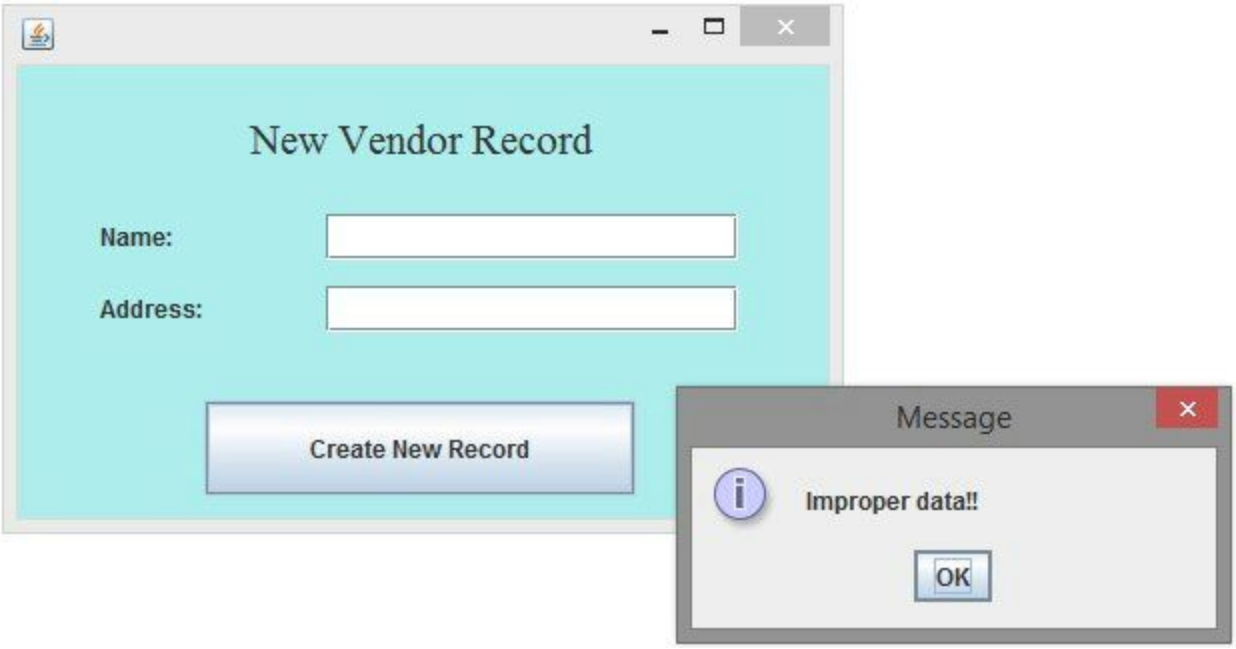
Expected system behaviour:

Acceptable data types : The system-generated vendor ID for the vendor is displayed. The shop-owner is informed via a message dialog box that the new record has been created.





Incorrect data types : An error message pops-up informing the user.



WHITE BOX TESTING : Checking that the database is correspondingly updated in the backend.

Before:

Vendor_ID	Name	Address
NULL	NULL	NULL

After:

Vendor_ID	Name	Address
Ho1	Howell	Islington
NULL	NULL	NULL

➢DISPLAYING RECORDS :

This feature was tested through black box testing only. On making a choice about the type of records to view, the user is presented with a display of the corresponding records with some of the details for each.

- DISPLAYING MEDICINE RECORDS :

Code_Number	Generic_Name	Trade_Name	Description	Quantity	Threshold_Value
AZ1	AZITHRAL	AZITHRAL	ANTIBIOTIC	90	5
NO2	NORFLOXACIN	NORFLOX	ANTIBIOTIC	5	5
P51	PARACETAMOL	P50	FEVER	0	5

- DISPLAYING VENDOR RECORDS :

Vendor_ID	Name	Address
A1	A.K. SAHA	TECH-M
BL3	BLUE PRINT	KOLKATA
DE2	DEVS MEDICAL	KHARAGPUR
FR4	FRANK ROSS	KOLKATA

➢UPDATING A RECORD:

○Updating a medicine record:

This test case condition arises during fresh stock arrival and sale of medicines.

BLACK BOX TEST FOR FRESH STOCK ARRIVAL : The shop-owner selects " Stock Arrival " from the main welcome page. A window pops up with the necessary fields to be updated. It also provides the option to print a cheque in favour of the supplier for payment for the stock.

Parameters : Medicine's Code Number, Batch Number, Vendor's ID, Per Unit Purchasing Price, Per Unit Selling Price, Expiry Date, Quantity of New Stock

System-acceptable values of the parameters:

- ➔ Medicine's code number : Code number of a medicine whose details already exist in the system's database.
- ➔ Batch Number : Batch number of the fresh stock of medicines arriving.
- ➔ Vendor's ID : The ID corresponding to some existing vendor record in the database.
- ➔ Per Unit Purchasing Price : Integer value as obtained depending on the medicine and the stock.

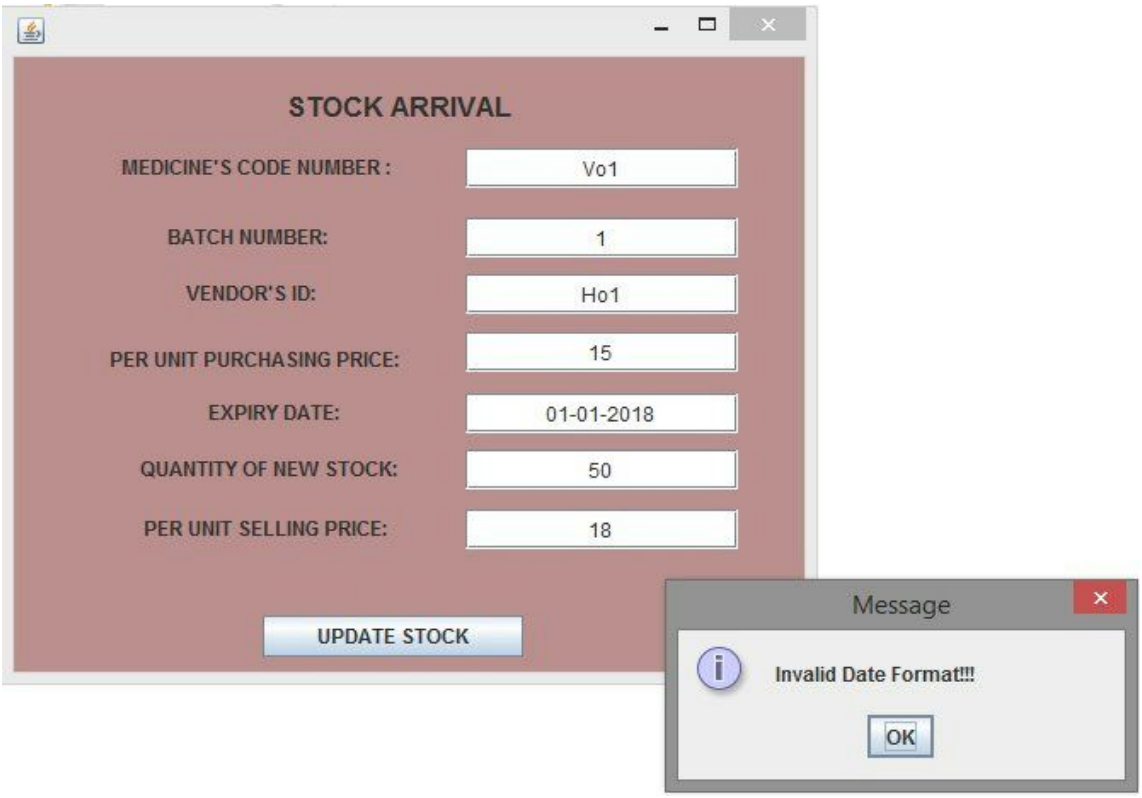
- ➔ Per Unit Selling Price : Integer value as specified depending on the medicine and the stock.
- ➔ Expiry Date : The date of expiry in parseable format for the fresh stock of medicines, assuming a particular stock has the same expiry date for all medicines of that stock.
- ➔ Quantity of New Stock : Integer value depending on the stock.

Expected system behaviour :

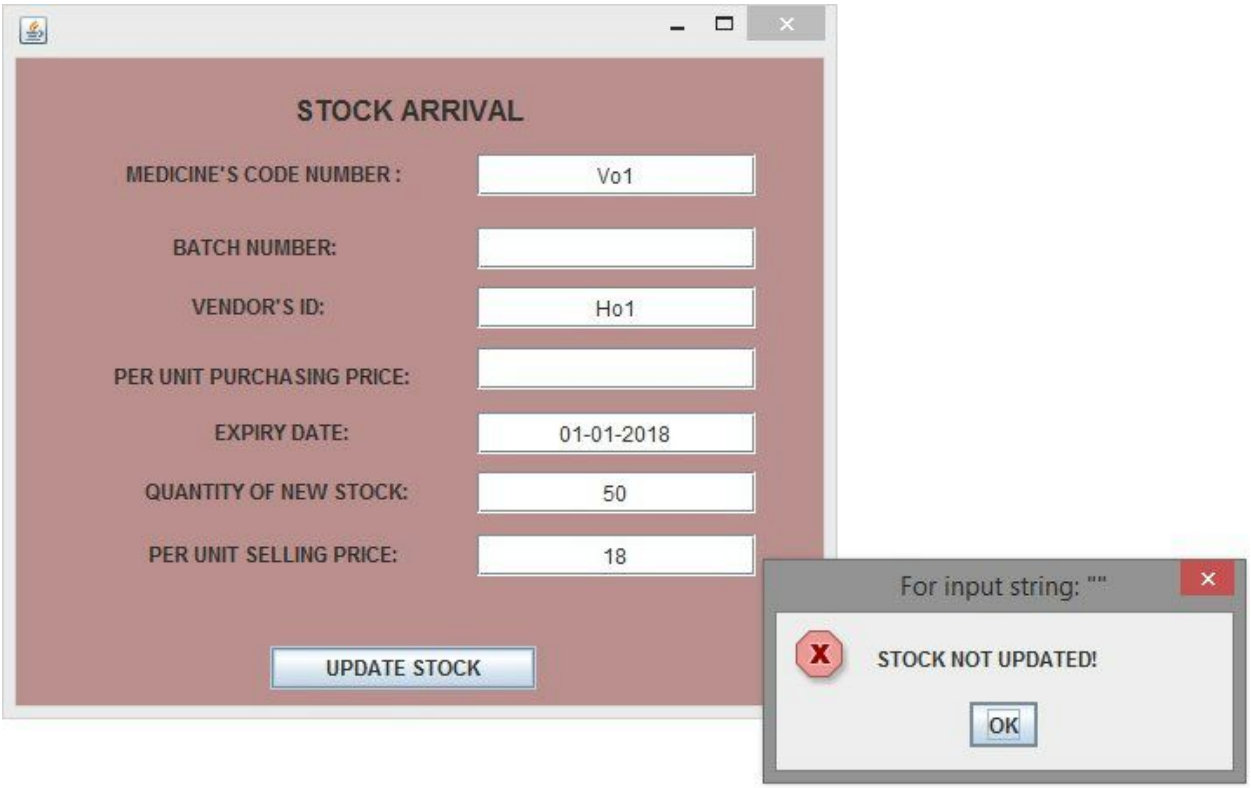
The shop-owner enters the details corresponding to all the parameters mentioned above. If the entered values correspond to the acceptable values, the database is updated. Thereafter, the shop-owner can print out a cheque in favour of the vendor to pay for the stock.

TEST CASES :

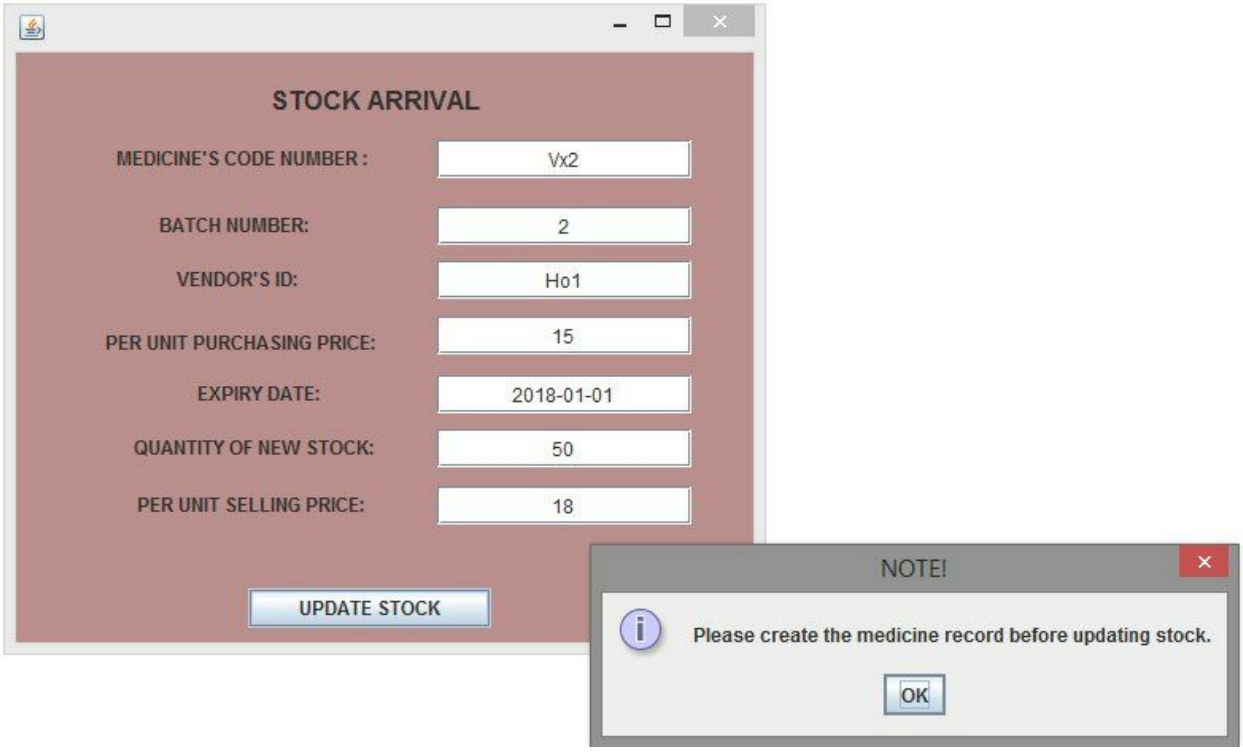
- **INVALID DATA TYPES :**
  - Invalid Date Format



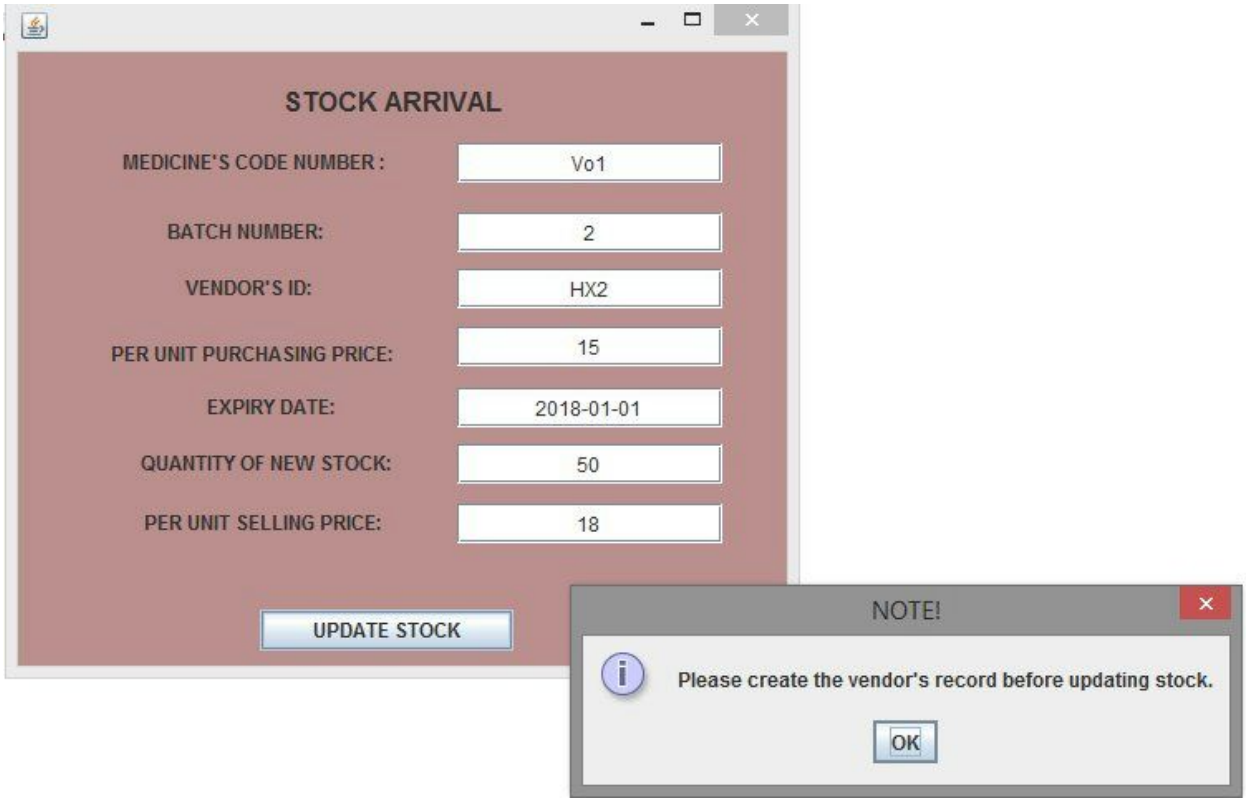
- Invalid Format for other input



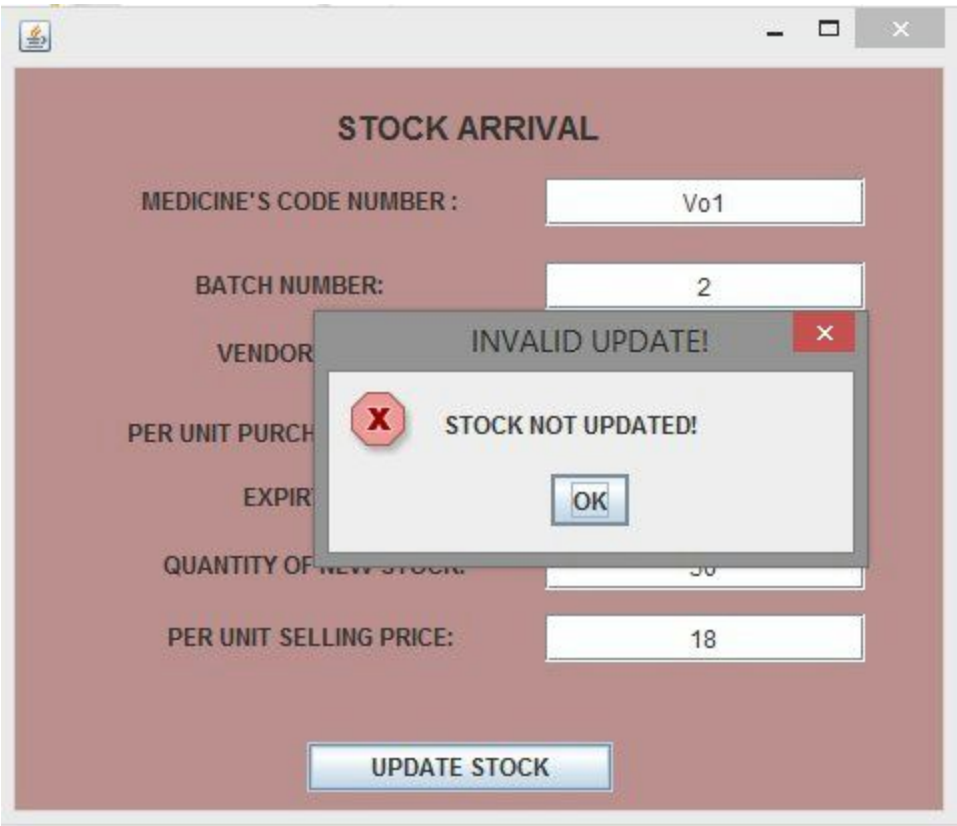
- ENTERED DETAILS CORRESPONDING TO SYSTEM-ACCEPTABLE PARAMETERS :
  - Non-existent medicine Record



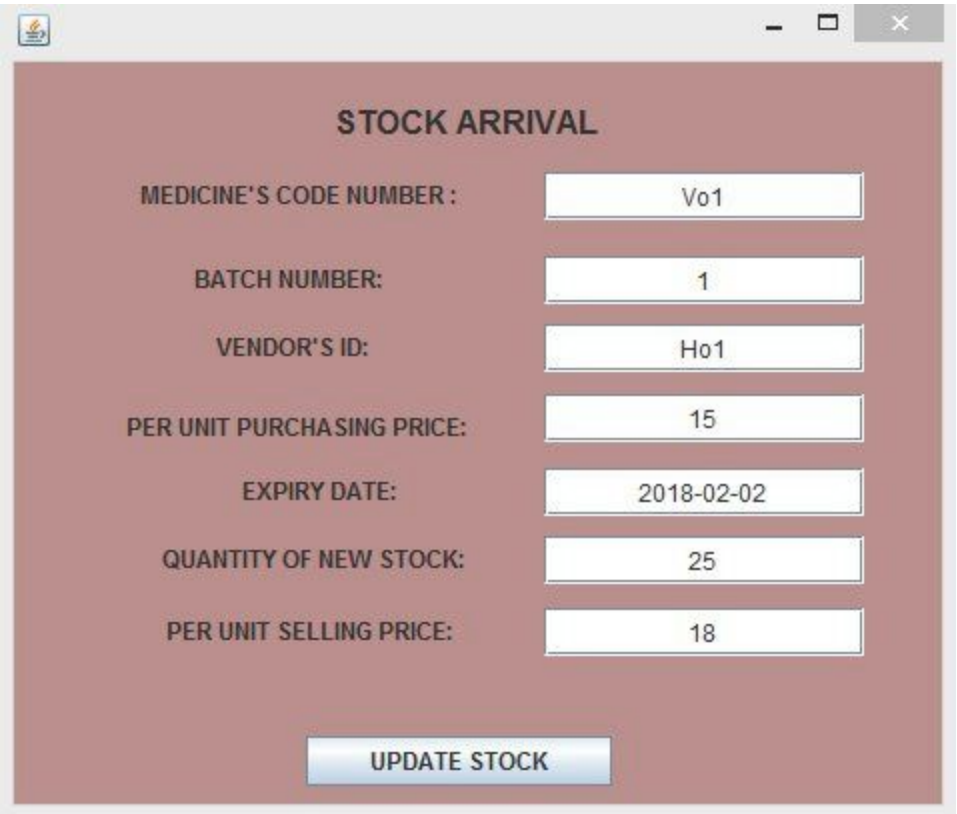
  - Non-existent vendor Record

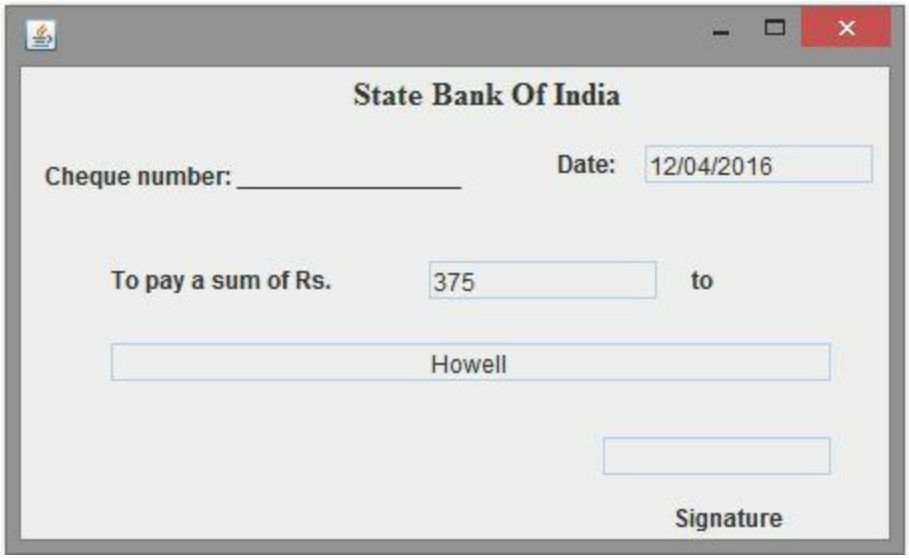


ERROR MESSAGE:



- **VALID INPUT:**


STOCK ARRIVAL
MEDICINE'S CODE NUMBER :
Vo1
BATCH NUMBER:
1
VENDOR'S ID:
Ho1
PER UNIT PURCHASING PRICE:
15
EXPIRY DATE:
2018-02-02
QUANTITY OF NEW STOCK:
25
PER UNIT SELLING PRICE:
18
UPDATE STOCK


State Bank Of India
Cheque number: _____
Date:
12/04/2016
To pay a sum of Rs.
375
to
Howell
Signature

WHITE BOX TEST FOR FRESH STOCK ARRIVAL :

```
//r is the resultset for querying the medicines database with the entered code number
if (! r.next()){ Alternate path when resultset r is empty (PATH 1)
      //no such record exists
      JOptionPane.showMessageDialog(null,"Please create the medicine record before updating stock.","NOTE!", JOptionPane.INFORMATION_MESSAGE);
      throw new Exception("Stock not updated!");
}else{ PATH WHEN RESULTSET NOT EMPTY (PATH 2)
      current_stock = r.getInt("Quantity");
}
try{
      Statement query = connect.createStatement();
      ResultSet result = query.executeQuery("SELECT * FROM vendor WHERE Vendor_ID = '"+txtVendorId.getText()+"'");
      result.beforeFirst();
      if (! result.next()){ ALTERNATE PATH WHEN VENDOR ID DOES NOT EXIST IN VENDOR DATABASE (PATH 3)
            JOptionPane.showMessageDialog(null,"Please create the vendor's record before updating stock.","NOTE!", JOptionPane.INFORMATION_MESSAGE);
            throw new Exception("INVALID UPDATE!");
      }
```

```
//execution of statements follow
      PreparedStatement statement = connect.prepareStatement("INSERT INTO stock (Code_Number, Batch_Number, Expiry_Date, Quantity, Per_Unit_Purchasing_Price, Per_Unit_Selling_Price,Vendor_ID, Stock_Arrival_Date) VALUES ('"+code_number+"', '"+txtBatchnumber.getText()+"', ? , "+Integer.parseInt(txtQuantity.getText())+", "+Integer.parseInt(txtPurchasingPrice.getText())+","+Integer.parseInt(txtSellingPrice.getText())+",'"+txtVendorId.getText()+"',? )");
      DateFormat format = new SimpleDateFormat("dd-MM-yyyy");
      Date date = new Date();
      java.sql.Date supply_date = new java.sql.Date(date.getTime());
      Date expiry_date = format.parse(txtExpiryDate.getText());
      java.sql.Date expiry = new java.sql.Date(expiry_date.getTime());
      statement.setDate(1, expiry);
      statement.setDate(2, supply_date);
      statement.executeUpdate();    STATEMENTS MARKED GREEN ARE CAPABLE OF THROWING EXCEPTIONS CAUGHT BY EXITING THE UPDATE ROUTINE
      System.out.println("Stock table updated!");
      try{
            PreparedStatement newstatement = connect.prepareStatement("UPDATE medicines SET Quantity = "+(current_stock + Integer.parseInt(txtQuantity.getText()))+" WHERE Code_Number = '"+code_number+"'");
            newstatement.executeUpdate();
            //System.out.println("Medicine table updated!");
      }catch(Exception e){ EXCEPTION HANDLING PATH (PATH 4)
            JOptionPane.showMessageDialog(null,"FAILED TO UPDATE medicines DATABASE","ERROR: ", JOptionPane.ERROR_MESSAGE);
            return;
      }
      //print_cheque();
      int price = Integer.parseInt(txtQuantity.getText()) * Integer.parseInt(txtPurchasingPrice.getText());
```

```
37.      Statement query1 = connect.createStatement();
38.      ResultSet  result1  =  query1.executeQuery("SELECT  *  FROM  vendor  WHERE
   Vendor_ID = '"+txtVendorId.getText()+"'");
39.      result1.beforeFirst();
40.
41.      if (result1.next()){
42.            Cheque.vendorname = result1.getString("Name");
43.            //System.out.println("vendorname set!");
44.      }
45.            Cheque.price = price;
46.            Cheque cheque = new Cheque();
47.            cheque.setVisible(true);//print the cheque
48.            //}
49.
50.      }catch(Exception e){ EXCEPTION HANDLING PATH (PATH 5)
51.            JoptionPane.showMessageDialog(null,"STOCK                    NOT
   UPDATED!",e.getMessage(),JOptionPane.ERROR_MESSAGE);
52.      }
53. }catch(Exception e){
      1. EXCEPTION HANDLING BY TERMINATING UPDATE ROUTINE (PATH 6)
   JOptionPane.showMessageDialog(null,e.getMessage(),"",JOptionPane.ERROR_MESSAGE);
54.      }
```

- ❑ <u>MEDICINE CODE NOT FOUND IN medicines DATABASE :</u> *Path 1 -> 6*
- ❑ <u>VENDOR ID NOT FOUND IN vendors DATABASE :</u> *Path 2 -> 3 -> 6*

- ❑ <u>FAILURE UPDATING THE DATABASE</u> : *Terminating at Path 4, 5 or 6*

Such paths are traversed by test cases involving invalid data entered, and other conditions leading to failure in querying the corresponding database.

<u>BLACK BOX TEST FOR SALE OF MEDICINE(S)</u> : The shop-owner selects " Sell Item " from the main welcome page. A window pops up with the necessary fields for input. It also provides the option to take a print out of the cash receipt to be given to the customer.

Parameters : Medicine Code Number, Batch Number, Units Sold

<u>System-acceptable values of the parameters:</u>

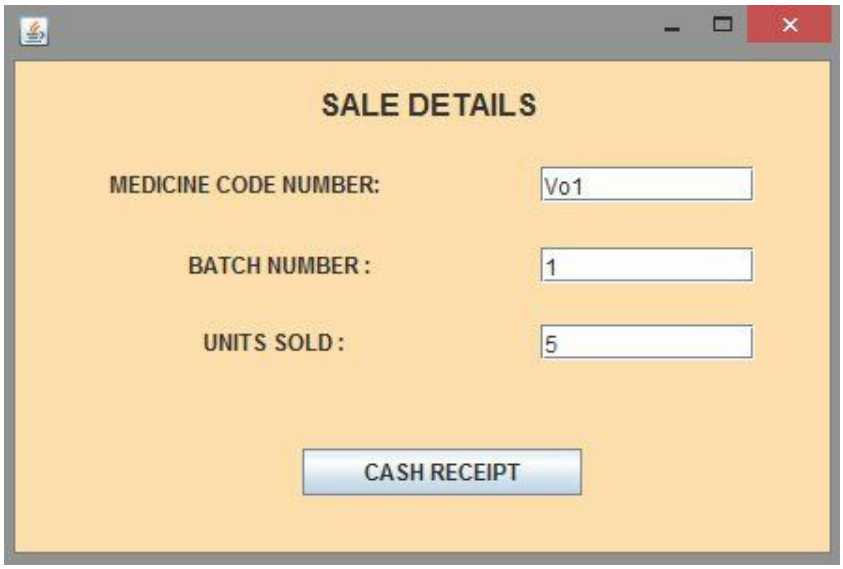
- ➔ Medicine Code Number : Code number of a medicine whose details already exist in the system's database.
- ➔ Batch Number : Batch number of the medicine(s) to sell.
- ➔ Units Sold : Integer value less than the current stock quantity as recorded in the medicines database.

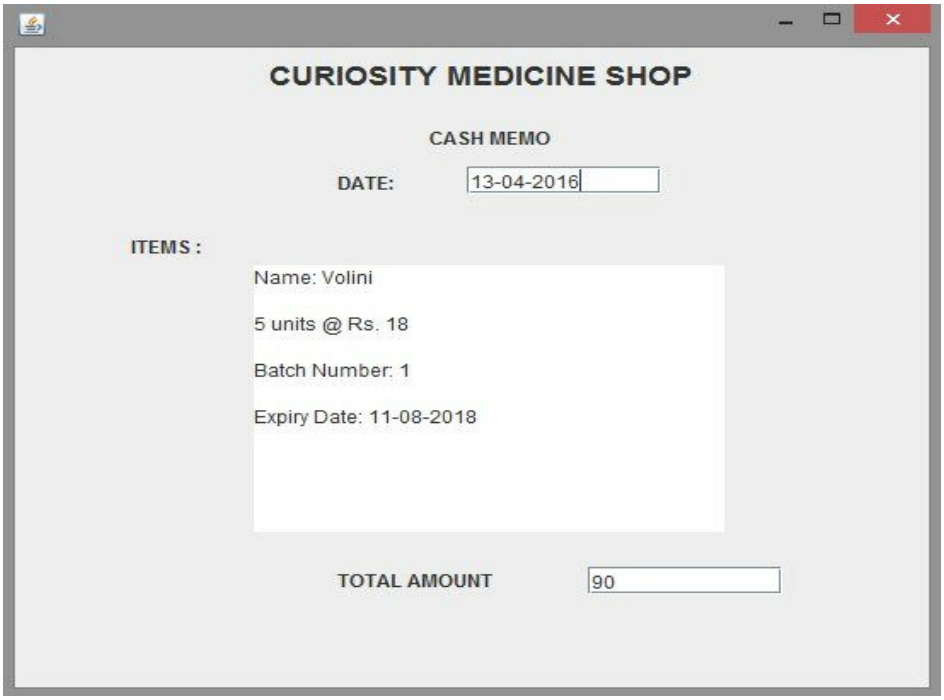
Expected system behaviour :

The shop-owner enters the details corresponding to all the parameters mentioned above. If the entered values correspond to the acceptable values, the database is updated. Thereafter, the shop-owner can print out the cash receipt for this transaction to be handed over to the customer.

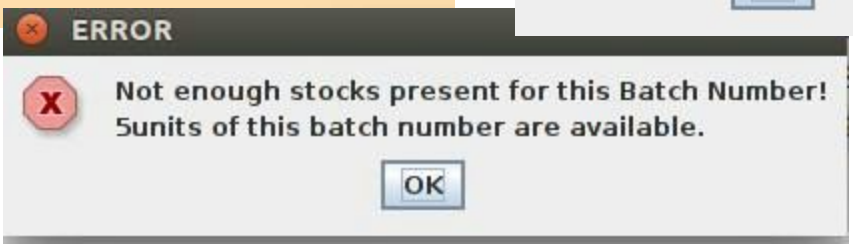
TEST CASES :

- ➔ DETAILS ENTERED CORRESPOND TO SYSTEM-ACCEPTABLE VALUES :
  - ◆ Valid quantity sold





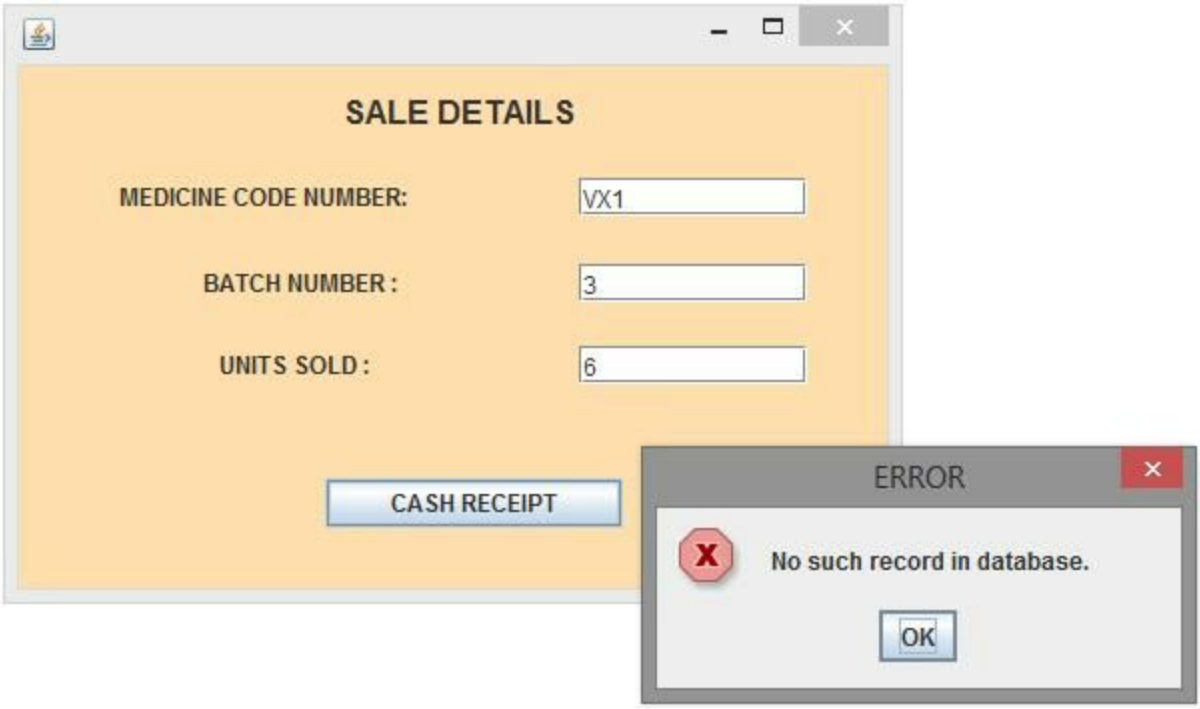
- <u>Invalid Quantity Sold</u>



The corresponding table in the database was verified for ensuring back-end validity.

#	Code_Number	Generic_Name	Trade_Name	Description	Quantity	Threshold_Value
1	AZ1	AZITHRAL	AZITHRAL	ANTIBIO...	90	5
2	NO2	NORFLOXACIN	NORFLOX	ANTIBIO...	5	5
3	P51	PARACETAMOL	P50	FEVER	0	5
*	NULL	NULL	NULL	NULL	NULL	NULL

➔ INVALID DETAILS :



❖ <u>WHITE BOX TEST FOR SALE OF MEDICINE(S)</u> :

```
//result is the resultset for querying the medicines database with the entered
code number
THOSE MARKED GREEN ARE POTENTIAL EXCEPTION HANDLING SCENARIOS FOR INVALID DATA
INPUT CONDITIONS OR UNSUCCESSFUL DATABASE QUERY
1. if (result.next()){ RECORD PRESENT IN DATABASE (PATH 1)
2.     //check if enough in stock to sell
3.     int current_quant = result.getInt("Quantity");
4.     if (current_quant < Integer.parseInt(txtUnitsSold.getText())){
5.           JOptionPane.showMessageDialog(null,"Not enough stocks present for
this Batch Number!\n"+ current_quant + "units of this batch number are
available." ,"ERROR",JOptionPane.ERROR_MESSAGE);
6.           throw new Exception(); (EXCEPTION PATH 2)
7.     }
8. int purchasing_price = result.getInt("Per_Unit_Purchasing_Price");
9. DateFormat format = new SimpleDateFormat("dd-MM-yyyy");
10. Date expiry_date =
   format.parse(format.format(result.getDate("Expiry_Date")));
11. java.sql.Date sql_expiry_date = new java.sql.Date(expiry_date.getTime());
12. int selling_price = result.getInt("Per_Unit_Selling_Price");
13. try{
14.    PreparedStatement statement = connect.prepareStatement("INSERT INTO sales
(Date_of_Sale, Code_Number, Batch_Number,Expiry_Date, Per_Unit_Selling_Price,
Quantity_Sold, Per_Unit_Purchasing_Price) VALUES
(?,'"+medicine_code+"','"+batch_number+"',?,"+selling_price+",
"+units_sold+" ,"+purchasing_price+")");
15.    statement.setDate(1, sql_date); //
16.    statement.setDate(2, sql_expiry_date); //
17.    statement.executeUpdate();
18.
19.    //System.out.println("Updated successfully!");
```

```
21.    try{
22.          PreparedStatement newstatement = connect.prepareStatement("UPDATE
stock SET Quantity = "+(current_quant - units_sold)+" WHERE ( Code_Number =
'"+medicine_code+"' AND Batch_Number = '"+batch_number+"')");
23.          newstatement.executeUpdate();
24.    //System.out.println("Stocks table Updated!");
25.
26.    Statement newquery = connect.createStatement();
27.    ResultSet medresult = newquery.executeQuery("SELECT * FROM medicines WHERE
Code_Number = '"+medicine_code+"'");
28.
29.    medresult.beforeFirst();
30.    if (medresult.next()){
31.          int total_stock = medresult.getInt("Quantity");
32.          int final_stock = total_stock - units_sold;
33.
34.          PreparedStatement updatestatement = connect.prepareStatement("UPDATE
medicines SET Quantity = "+final_stock+" WHERE Code_Number =
'"+medicine_code+"'");
35.          updatestatement.executeUpdate();
36.          //System.out.println("medicines table updated!");
37.    }
38.    String trade_name = medresult.getString("Trade_Name");
```

```
39.    CashReceipt.name = trade_name;
40.    CashReceipt.unit_price = selling_price;
41.    CashReceipt.expiry = expiry_date;
42.
43.    //if stocks updated also, finally print cash receipt for the customer
44.    CashReceipt receipt = new CashReceipt();
45.    receipt.setVisible(true);
46.    }catch(Exception e){
47.          JOptionPane.showMessageDialog(null,"Could not update medicine
stocks' database!","ERROR",JOptionPane.ERROR_MESSAGE);
48.    }
49.    }catch(Exception e){
50.          JOptionPane.showMessageDialog(null,"Could not update sales'
database!","ERROR",JOptionPane.ERROR_MESSAGE);
51.          //System.out.println(e);
52.    }
53.    }else{ ELSE CONDITION CORRESPONDING TO PATH 1
54.          JOptionPane.showMessageDialog(null,"No such record in
database.","ERROR",JOptionPane.ERROR_MESSAGE);
55.    }
56.
57.    }catch(Exception e){ RETURN PATH FOR UNSUCCESFUL TRANSACTION (PATH 3)
58.          JoptionPane.showMessageDialog(null,"SALE
ABORTED!","ERROR",JOptionPane.ERROR_MESSAGE);
59. }
```

- <u>MEDICINE CODE NOT FOUND IN medicines DATABASE :</u> *ELSE CONDITION CORRESPONDING TO PATH 1*
- <u>NOT ENOUGH STOCKS PRESENT TO SELL</u> : *EXCEPTION PATH 2 -> RETURN PATH 3*
- <u>FAILURE UPDATING THE DATABASE</u> : *Terminating at either of the try-catch blocks highlighted or resulting in unsuccessful transaction via path 3*

Such paths are traversed by test cases involving invalid data entered, and other conditions leading to failure in querying the corresponding database.

➢<u>SEARCH:</u>

This test case condition arises when the shop-owner decides to search the details of a medicine record from the shop's database.

<u>BLACK BOX TEST FOR SEARCHING A MEDICINE RECORD</u> : The shop-owner enters the required medicine's Trade name or Generic name (whichever is prefered) in the appropriate marked field in the main welcome page.

Parameters : Trade Name or Generic Name

<u>System-acceptable values of the parameters</u>:

➔ Trade Name : Trade name of a medicine whose details already exist in the system's database.

➔ Generic Name : Generic name of a medicine whose details already exist in the system's database.

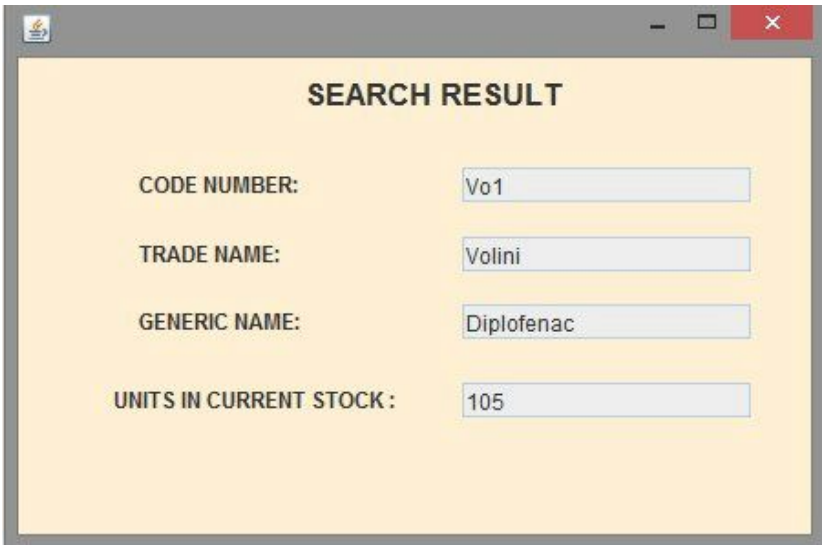
Expected system behaviour :

The shop-owner enters the Trade name or Generic name of the medicine that he/she wishes to delete. If the entered value corresponds to an acceptable value, the database is updated.

TEST CASES :

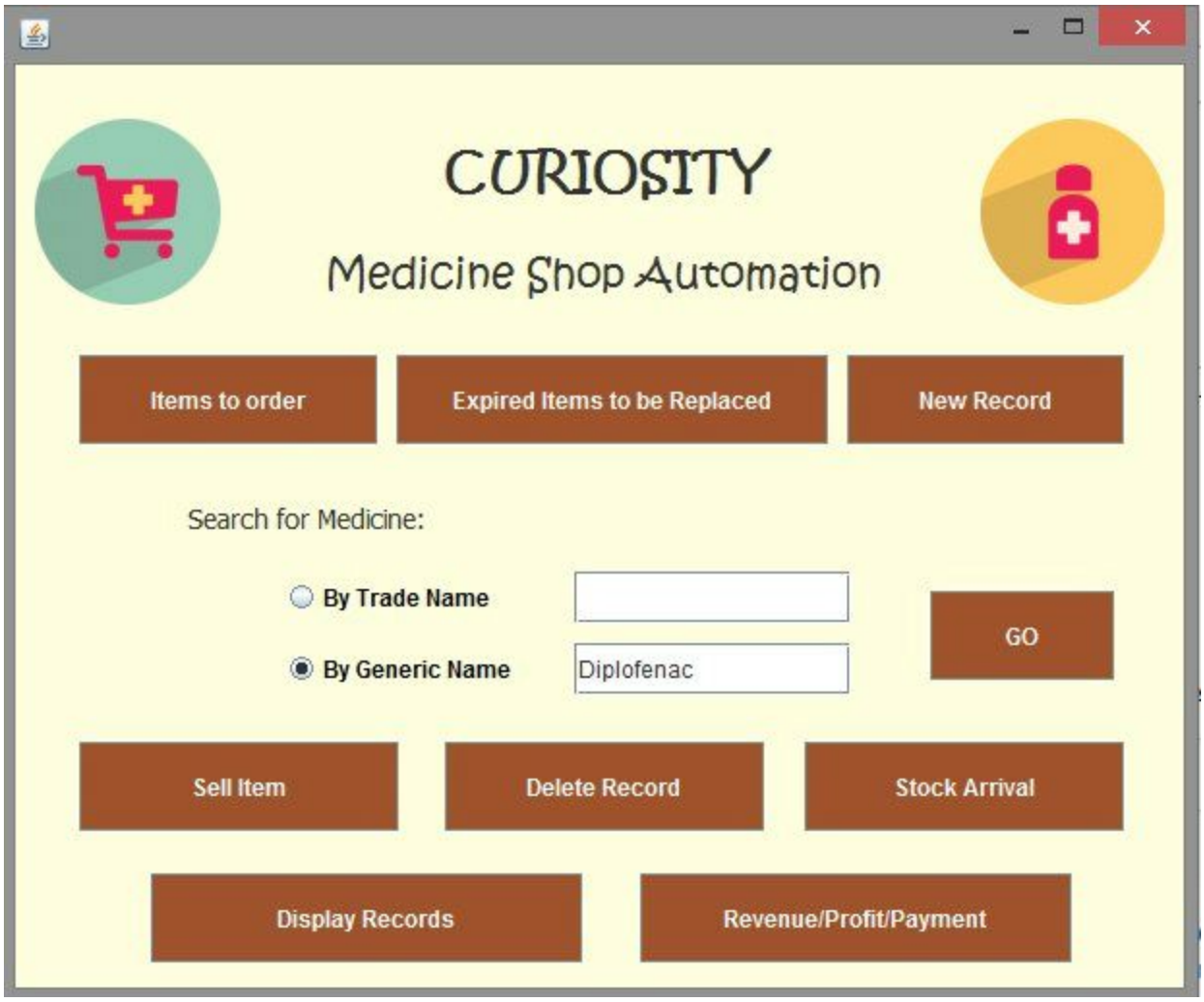
● INVALID TRADE NAME:

● VALID TRADE NAME:



● INVALID GENERIC NAME:

● VALID GENERIC NAME:



➢QUERIES:

This involves the test case conditions for generating revenue, profit and vendor-wise payment figures for a certain period, as well as obtaining the list of expired medicines to be replaced.

BLACK BOX FOR OBTAINING THE LIST OF ITEMS TO ORDER:

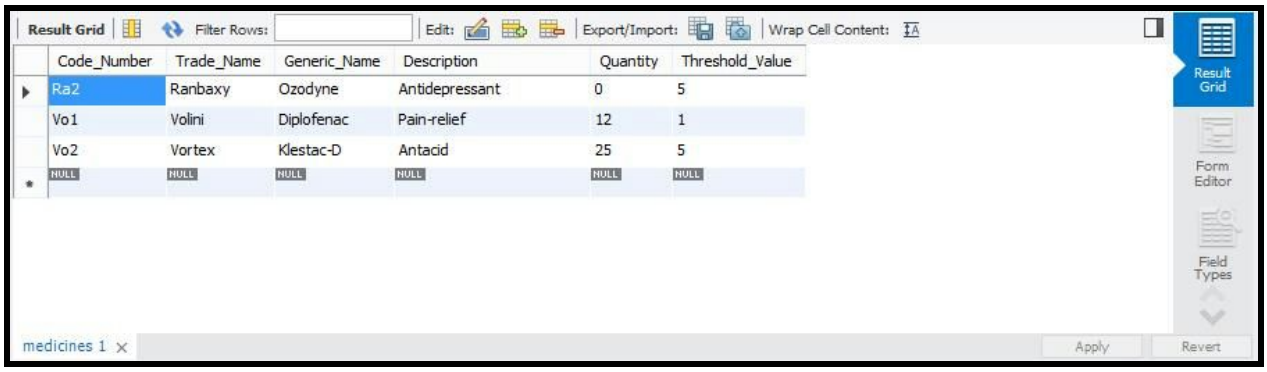
The shop-owner selects " Show Items To Order" option from the main Welcome page to view the items whose quantity is lesser than their threshold value.

Expected system behaviour :

The MSA System computes the threshold value of each medicine record and updates the database with the information. It then prints out a list of items that have their threshold values more than the quantity present.

Test Case:

Medicines Table (Back-end validation):



Code_Number	Trade_Name	Generic_Name	Description	Quantity	Threshold_Value
Ra2	Ranbaxy	Ozodyne	Antidepressant	0	5
Vo1	Volini	Diplofenac	Pain-relief	12	1
Vo2	Vortex	Klestac-D	Antacid	25	5
NULL	NULL	NULL	NULL	NULL	NULL

Sales Table (Back-end validation):

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Date_of_Sale	Code_Number	Batch_Number	Expiry_Date	Per_Unit_Selling_Price	Quantity_Sold	Per_Unit_Purchasing_Price
2016-04-13	Vo1	1	2018-08-11	18	5	15
NULL	NULL	NULL	NULL	NULL	NULL	NULL

sales 1 | Apply | Revert

Output(Front-end):

Code_Number	Trade_Name	Generic_Name	Description	Quantity	Threshold_Value	Quantity_To_Order
Ra2	Ranbaxy	Ozodyne	Antidepressant	0	5	20

<u>BLACK BOX TEST FOR OBTAINING THE LIST OF EXPIRED ITEMS</u> :

The shop-owner selects " Show Expired Items" option from the main Welcome page to view the medicines that have already expired.

<u>Expected system behaviour</u> :

The MSA System compares the expiry date of each medicine with the present date and then prints out the list of expired items.

<u>Test Case</u> -

<u>Medicines Table(Back-end validation)</u> -

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Code_Number	Trade_Name	Generic_Name	Description	Quantity	Threshold_Value
Ra2	Ranbaxy	Ozodyne	Antidepressant	40	12
Vo1	Volini	Diplofenac	Pain-relief	50	13
Vo2	Vortex	Klestac-D	Antacid	100	20
NULL	NULL	NULL	NULL	NULL	NULL

medicines 1 | Apply | Revert

Stock Table (Back-end validation)-

Code_Number	Batch_Number	Expiry_Date	Quantity	Per_Unit_Purchasing_Price	Per_Unit_Selling_Price	Stock_Arrival_Date	Vendor_ID
Vo1	1	2018-08-11	25	15	18	2016-04-12	Ho1
Vo2	1	2022-05-06	25	15	18	2016-05-04	Le3
Vo1	2	2016-04-15	25	15	18	2012-02-02	Ho1
Vo2	2	2016-02-12	75	15	18	2012-05-01	Ho1
Ra2	5	2014-01-01	40	30	32	2011-01-10	Le1
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

RESULT (Front-end):

Code_Number	Trade_Name	Generic_Name	Description	Quantity	Threshold_Value
Vo2	Vortex	Klestac-D	Antacid	25	20
Ra2	Ranbaxy	Ozodyne	Antidepressant	0	12

BLACK BOX TEST FOR GENERATING THE REVENUE, PROFIT AND VENDOR-WISE PAYMENT FIGURES FOR A CERTAIN PERIOD :

The shop-owner selects " Revenue/Profit/Payment " from the main welcome page. The window that pops up asks the user to select a start date and an end date corresponding to which the system can generate the revenue or profit or vendor-wise payment data for that duration.

PARAMETERS : Start Date, End Date

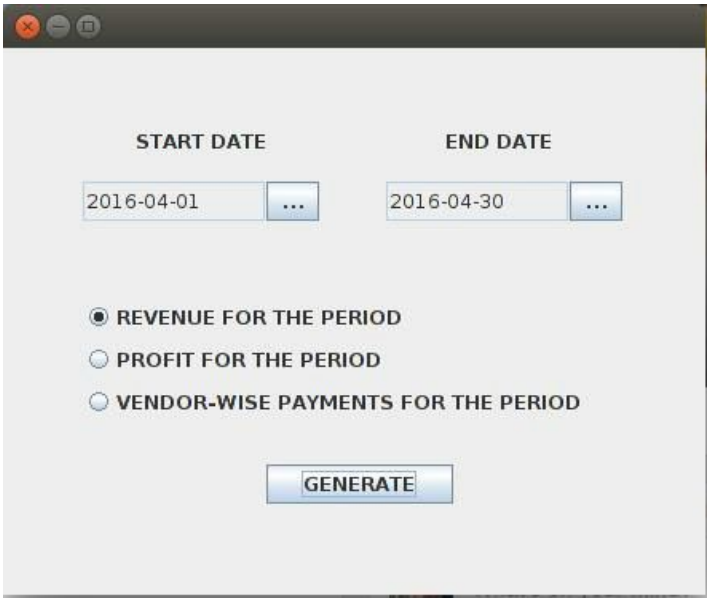
Expected System Behaviour :

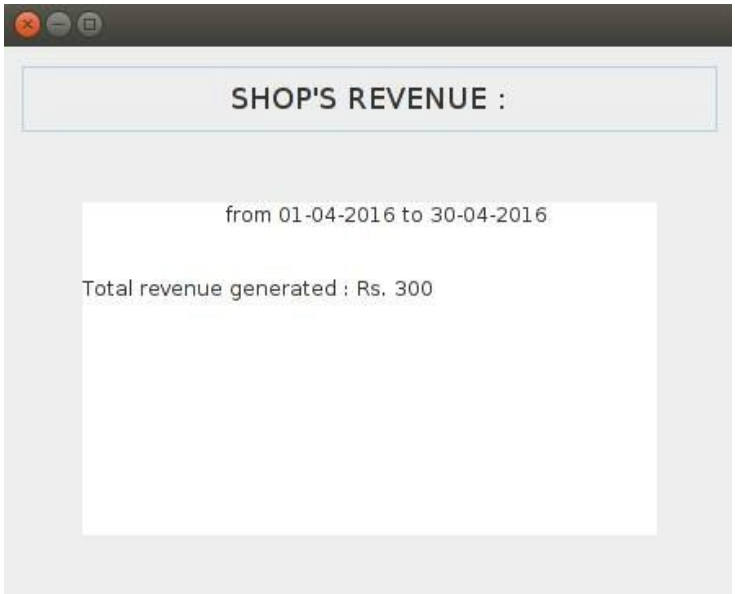
- ➔ The start date and end date selected if valid (i.e. start date before or same as end date) the system generates the data as requested.

➔ The request is one of the options to generate revenue, profit or vendor-wise payment figures for the specified duration.

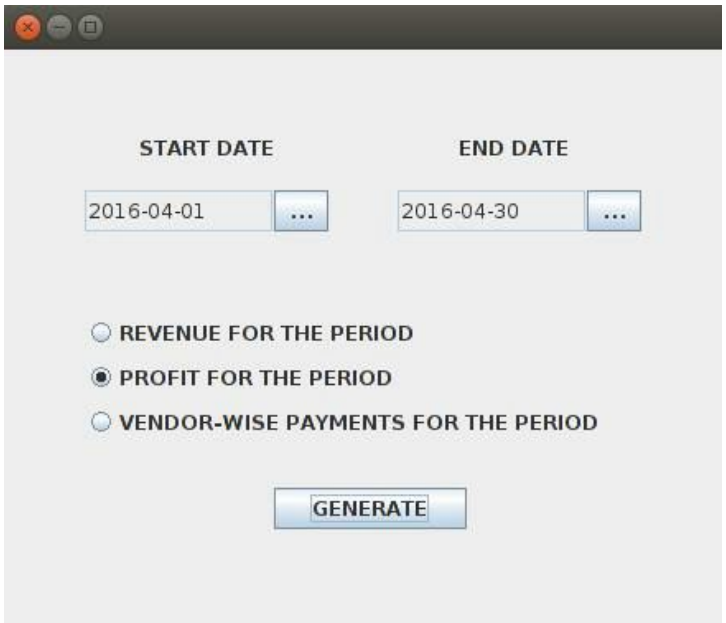
TEST CASES :

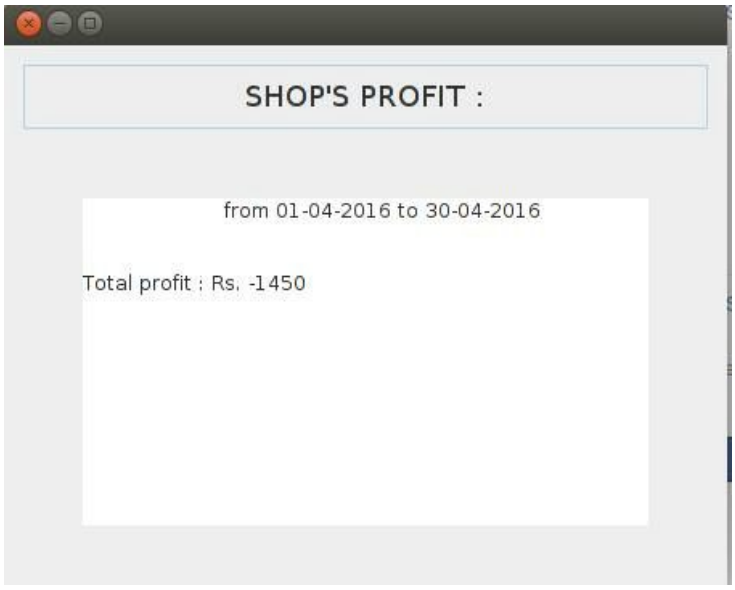
- GENERATE REVENUE DATA :



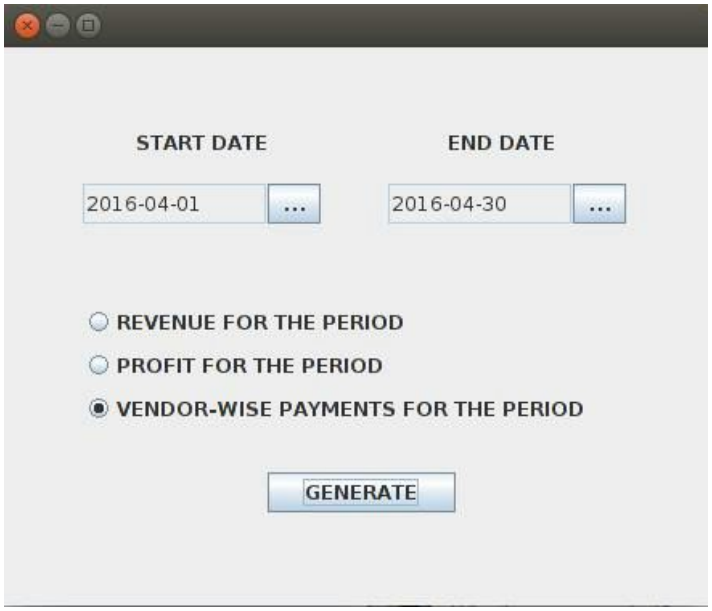


- GENERATE PROFIT DATA :

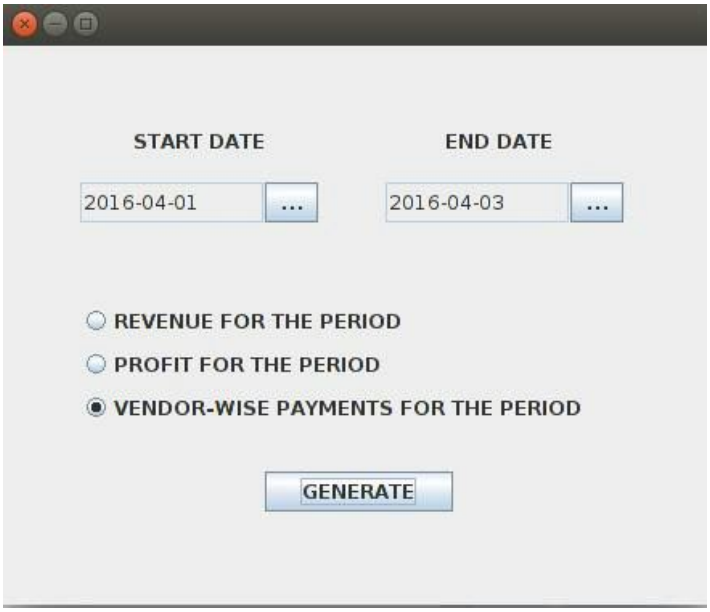




- GENERATE VENDOR-WISE PAYMENT DATA :



- NO RECORDS FOR QUERIED PERIOD :



STOCK TABLE (Back-end Validation):

#	Code_Number	Batch_Number	Expiry_Date	Quantity	Per_Unit_Purchasing_Price	Per_Unit_Selling_Price	Stock_Arrival_Date	Vendor_ID
1	AZ1	SN23B	2017-06-03	40	10	12	2016-04-06	A.1
2	AZ1	SN23B	2017-06-03	40	20	12	2016-04-06	BL3
3	NO2	MS23QP	2017-08-04	50	10	12	2016-04-06	BL3
4	AZ1	25FG67	0016-03-10	20	25	29	2016-04-12	BL3

SALES TABLE (Back-end Validation):

#	Date_of_Sale	Code_Number	Batch_Number	Expiry_Date	Per_Unit_Selling_Price	Quantity_Sold	Per_Unit_Purchasing_Price
1	2016-04-13	AZ1	SN23B	2017-06-03	12	10	10
2	2016-04-13	NO2	MS23QP	2017-08-04	12	15	10

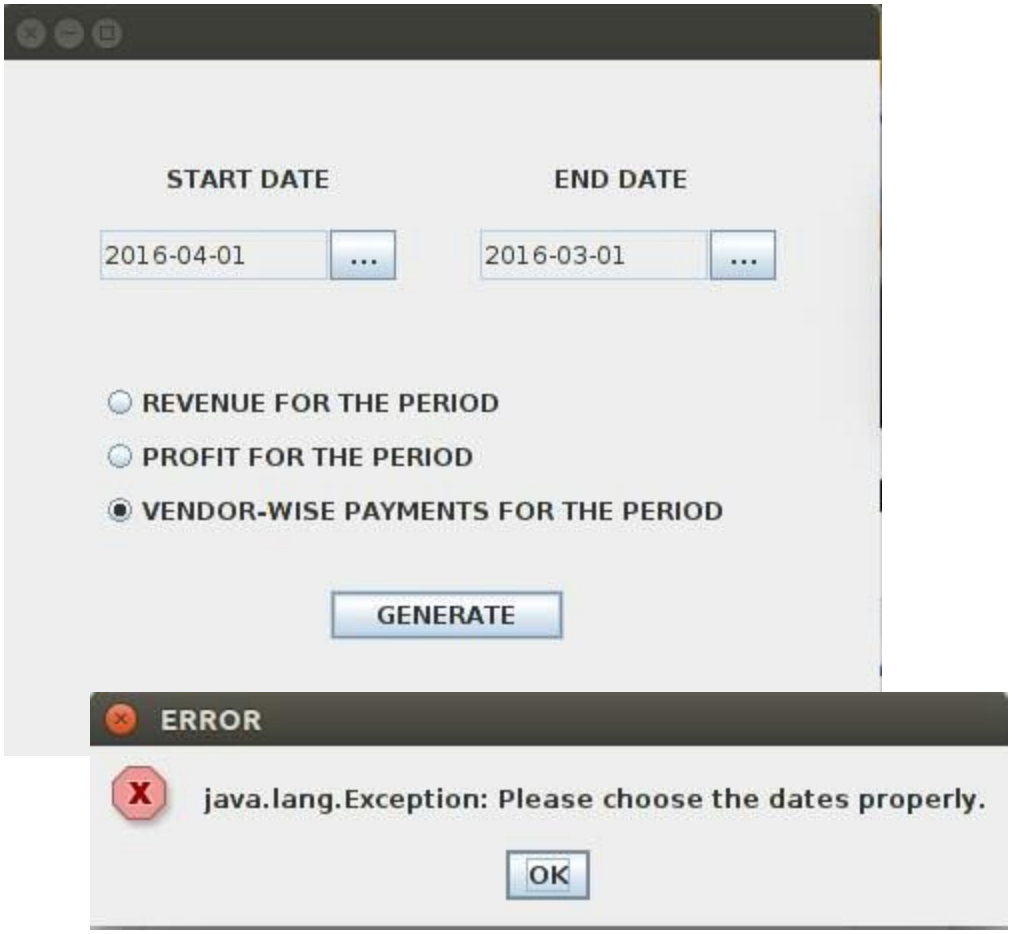
<u>WHITE BOX TEST FOR GENERATING THE REVENUE, PROFIT AND VENDOR-WISE PAYMENT FIGURES FOR A CERTAIN PERIOD :</u>

```
1.  try{
2.      Date startdate = (Date) startdatePicker.getModel().getValue();
3.      Date enddate = (Date) enddatePicker.getModel().getValue();
4.      if (enddate.before(startdate)){ EXCEPTION PATH 1
5.          throw new Exception("Please choose the dates properly.");
6.      }
7.      else if (enddate.after(startdate) || enddate.equals(startdate)){
8.          if (rdbtnRevenue.isSelected()){
9.              generate_revenue();
10.         }
11.         else if (rdbtnProfit.isSelected()){
12.             generate_profit();
13.         }
14.         else if (rdbtnVendor.isSelected()){
15.         //generate vendor-wise payments
16.             generate_vendorwise_payments();
17.         }
18.     }
19.
20.     }catch(Exception e){ TERMINATE PROCESS PATH 2
21.         JOptionPane.showMessageDialog(null, e,"ERROR",
JOptionPane.ERROR_MESSAGE);
22.     }
23.
```

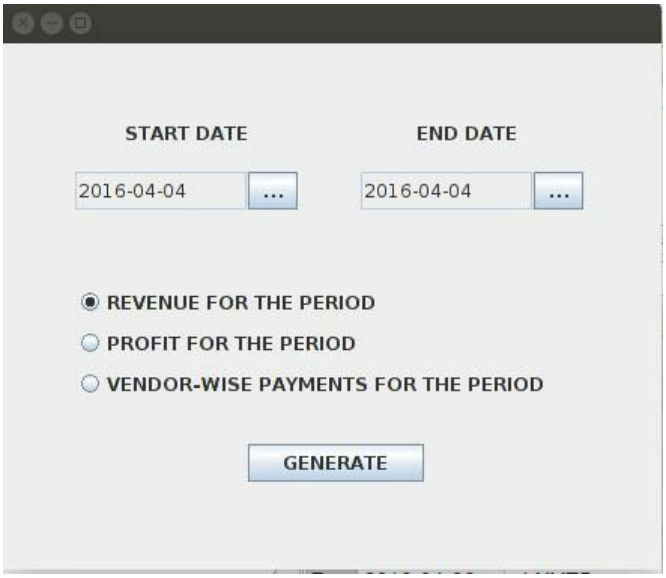
- ➔ Depending on whether the selected start and end dates are appropriate, the system covers either the path 1 -> 2 or the path corresponding to the else condition of path 1.
- ➔ As a check for boundary conditions, a test case having the same start and end date can be used to generate and observe the output.

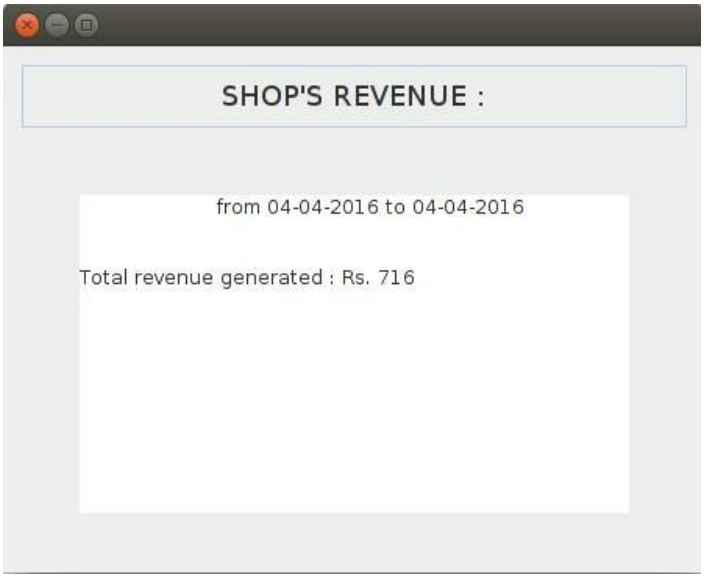
➔ The subsequent queries on different paths i.e. to generate revenue, profit and vendor-wise payment figures will return desired results if the corresponding record exists or will exit the process in case of a failed query.

➔ <u>EXCEPTION PATH 1 FOLLOWED</u> :



➔ NO EXCEPTION RAISED IF START DATE SAME AS END DATE :





➢DELETING A RECORD:

○ Deleting a medicine record:

This test case condition arises when the shop-owner decided to delete a medicine record from the shop's database.

BLACK BOX TEST FOR DELETING A MEDICINE RECORD : The shop-owner selects " Delete Record" from the main welcome page. A window pops up with the choices of medicine or vendor records. The user makes the "Delete a medicine record" choice from the drop-down list of options.

Parameters : Medicine's Code Number

System-acceptable values of the parameters:

- ➔ Medicine's code number : Code number of a medicine whose details already exist in the system's database.

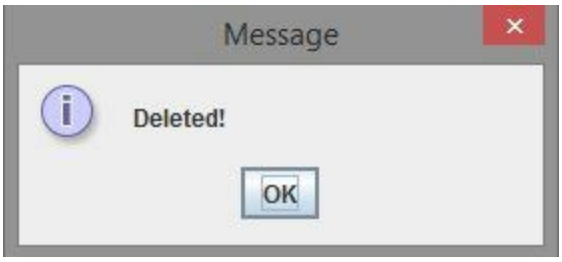
Expected system behaviour :

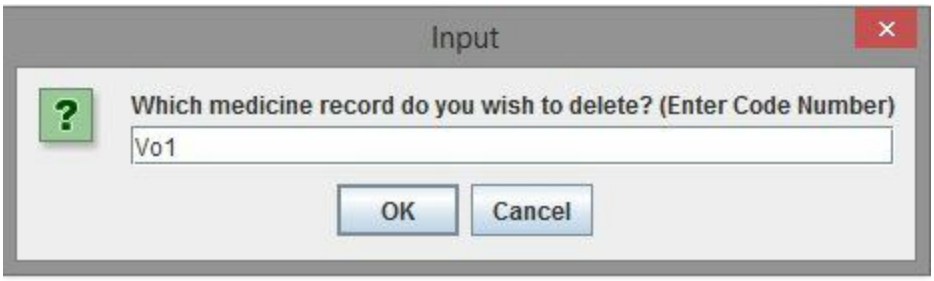
The shop-owner enters the code number of the medicine that he/she wishes to delete. If the entered value corresponds to an acceptable value, the database is updated.

TEST CASES :

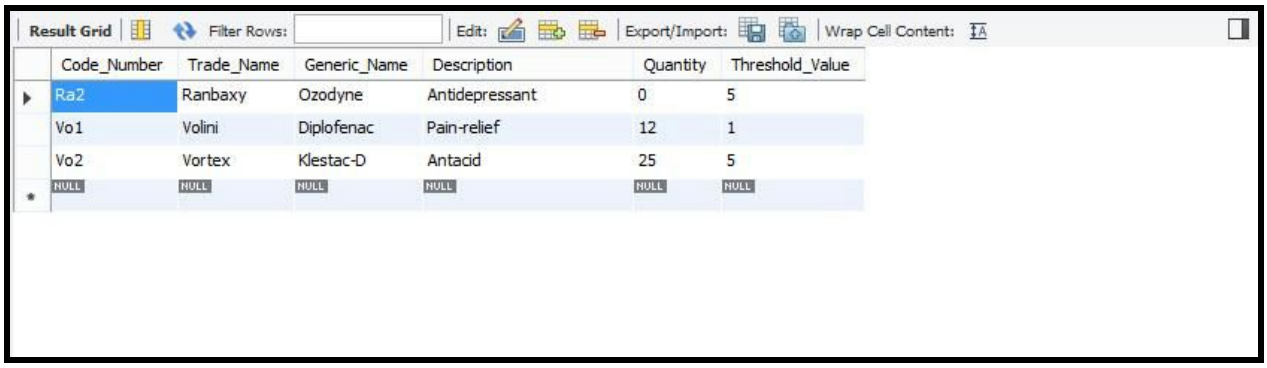
- INVALID CODE NUMBER

- VALID CODE NUMBER



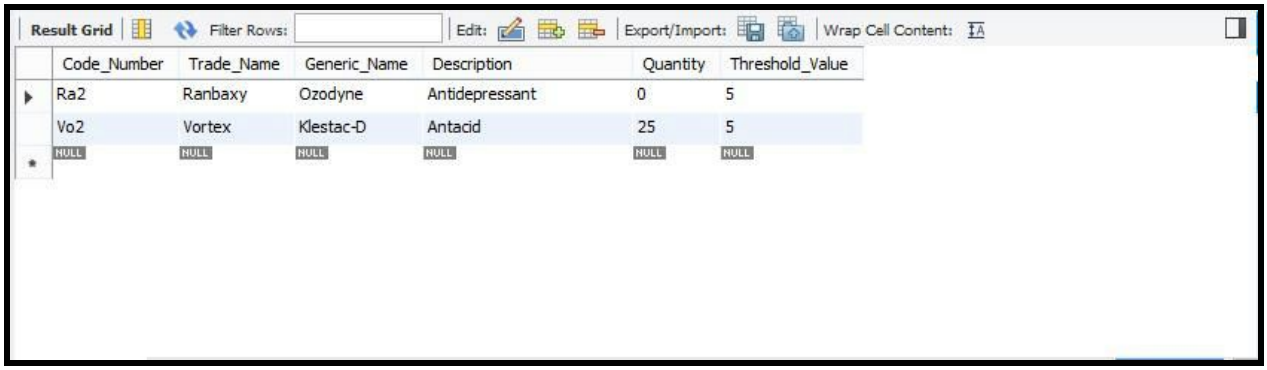


Checking the corresponding table in the database before deletion:



Code_Number	Trade_Name	Generic_Name	Description	Quantity	Threshold_Value
Ra2	Ranbaxy	Ozodyne	Antidepressant	0	5
Vo1	Volini	Diplofenac	Pain-relief	12	1
Vo2	Vortex	Klestac-D	Antacid	25	5
NULL	NULL	NULL	NULL	NULL	NULL

After deletion of record:



Code_Number	Trade_Name	Generic_Name	Description	Quantity	Threshold_Value
Ra2	Ranbaxy	Ozodyne	Antidepressant	0	5
Vo2	Vortex	Klestac-D	Antacid	25	5
NULL	NULL	NULL	NULL	NULL	NULL

○ Deleting a vendor record:

This test case condition arises when the shop-owner decided to delete a vendor's record from the shop's database.

BLACK BOX TEST FOR DELETING A VENDOR RECORD : The shop-owner selects " Delete Record" from the main welcome page. A window pops up with the choices of medicine or vendor records. The user makes the "Delete a vendor record" choice from the drop-down list of options.

Parameters : Vendor's ID

System-acceptable values of the parameters:

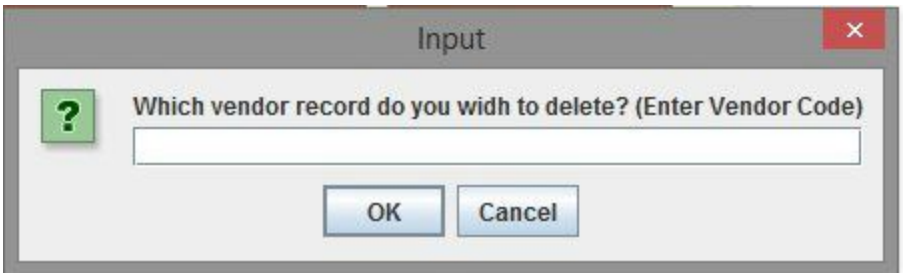
➔ Vendor ID : The unique Identity number of a vendor whose details already exist in the system's database.

Expected system behaviour :

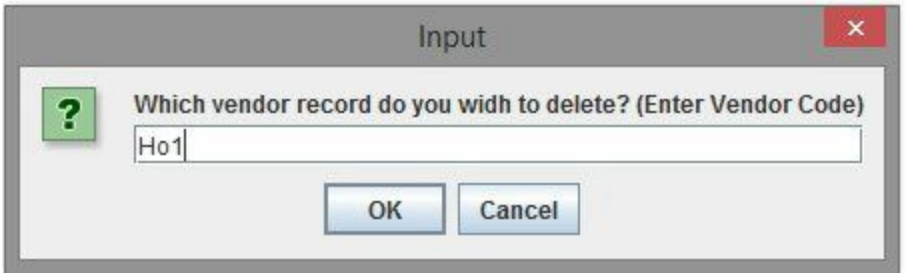
The shop-owner enters the Vendor ID of the vendor that he/she wishes to delete. If the entered value corresponds to an acceptable value, the database is updated.

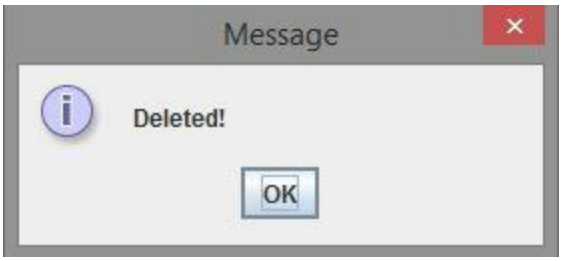
TEST CASES :

- INVALID VENDOR ID



- VALID VENDOR ID

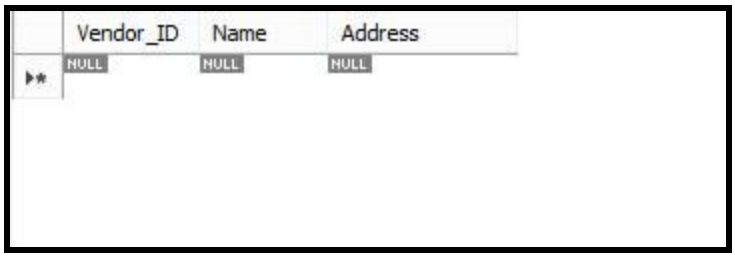




Database table before deletion:

Vendor_ID	Name	Address
Ho1	Howell	Islington
NULL	NULL	NULL

Database table after deletion:



Vendor_ID	Name	Address
NULL	NULL	NULL

❑ SYSTEM TESTING :-

The goals of system testing are to detect faults that can only be exposed by testing the entire integrated system or some major part of it. Generally, system testing is mainly concerned with areas such as performance, security, validation, load/stress, and configuration sensitivity. But in our case we focused only on function validation and performance.

➔ FUNCTION VALIDATION TESTING :

All the functionalities mentioned in the SRS document were individually tested. This was primarily a UI based testing. It was also verified that the databases were appropriately updated in the backend.The functions tested were:

- ❖ Login
- ❖ Creating a new record
- ❖ Displaying records in the database
- ❖ Sales transaction
- ❖ Stock arrival and updating
- ❖ Generating list of expired items
- ❖ Generating list of items to be ordered
- ❖ Querying the database for medicine record
- ❖ Generating period based revenue, profit and vendor-wise payment data
- ❖ Deleting record from the database

➔ PERFORMANCE TESTING :

This test was conducted to evaluate the fulfillment of a system with specified performance requirements.Following things were tested:

- ◆ Adding large number of medicine records to the database to see how much time it takes to retrieve them from the server when queried.
- ◆ Performing a large number of transactions to see whether all the tables in the database were appropriately updated.
- ◆ Calculating the sales statistics for a very large sales history to test the performance of revenue and profit generation.
- ◆ Performing a large number of transactions to check whether the software could generate the quantity of items to be ordered by calculating the expected values from past sales figures, thereby complying with the original goal of JIT method.
- ◆ Adding a large number of stock arrival events to check whether the stocks were appropriately updated and the vendor-wise payment figures appropriately calculated.