Criterion A: Planning

Transaction Management System

<u>Identifying the Problem</u>

My client is the personnel of the trading sector in a small business. She keeps track of the transactions of medical equipment purchased from different suppliers, which is then sold and distributed to various buyers. Her issue is that it "takes some time to find all the details" (see Appendix 1). My client is looking for a faster and more organized method to manage the transactions than the current paper-document system.

Because medical equipment suppliers can be more or less reliable they are divided into three categories based on previous transactions. Reliable providers are in group "C" and new providers are in group "A". Whenever my client needs to check information about any of the transactions, she has to find the correct documents in paper form. After talking to the client I realized that the problem lies in the accessibility of the transactions data and that there are no hard rules for managing this data. That leads to the main problem: Although a paper record of the transactions is present, the current process is slow and inefficient. It also requires my client to use physical copies of the documents.

Word count: 189

<u>Rationale for the Solution:</u>

I decided to design a solution, which would improve the functioning of the transaction record system for my Internal Assessment in CS. The user of my product (my client) should be able to view the transactions from suppliers of medical equipment as well the suppliers themselves. My program should allow the user to keep track of the current transactions, input new ones and store them between sessions. Additionally, a sortable list of suppliers should be available to the user.

The aim of my product is to improve the efficiency of the transaction management for the client. The old paper-document system is slow and not very effective, hence I will create a database management system with a clear user interface that automates the time-consuming tasks.

I will be using SQLite database to store the transactions data along with Java NetBeans IDE to design the GUI and to write the logic of my program. I chose Java because I am familiar with it and because it is a powerful programming language that supports objects and allows to create GUI interfaces. It can work with databases, such as SQLite.

NetBeans IDE provides all necessary tools (such as Swing) to create a fully functional application including the graphical interface. It has great java support and is relatively easy to use. Use of a relational database will allow me to store information about the suppliers, medical equipment, and most importantly – transactions without repeating this information each time it needs to be used in a table. This will save storage, make the program more efficient and allow for large number of data records.

Word count: 267

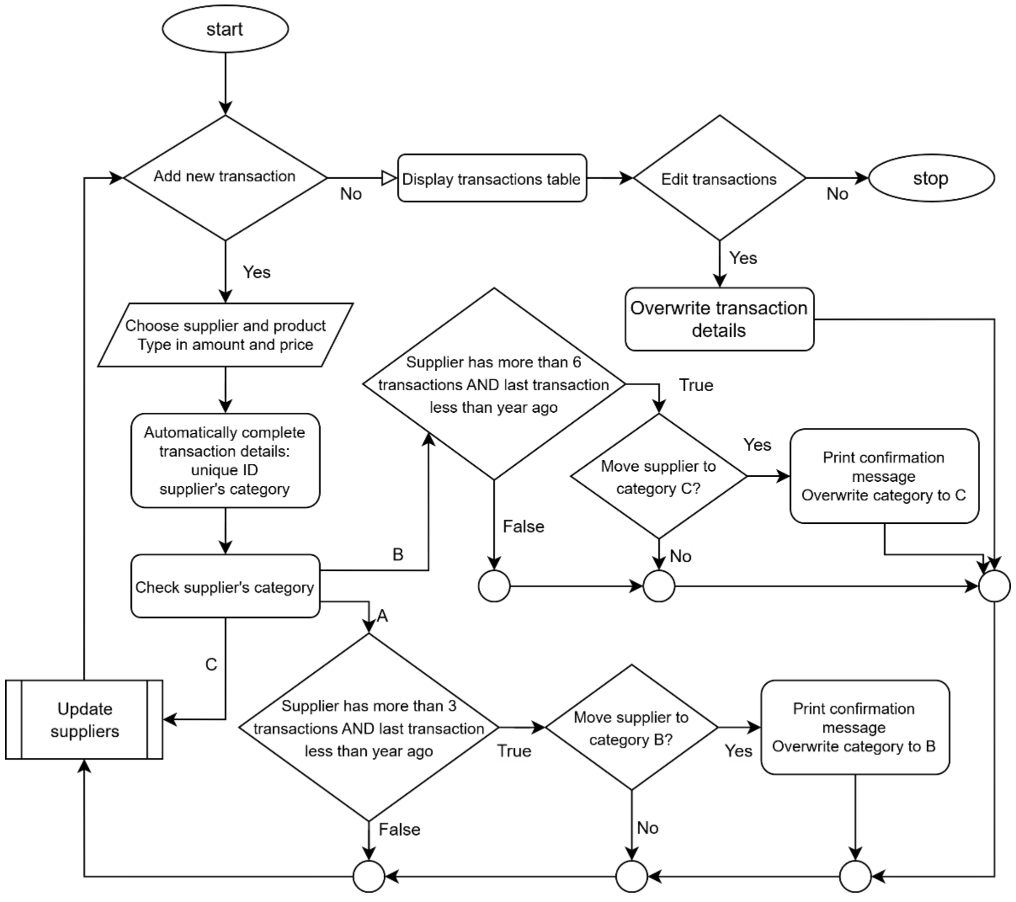
<u>Success criteria:</u>

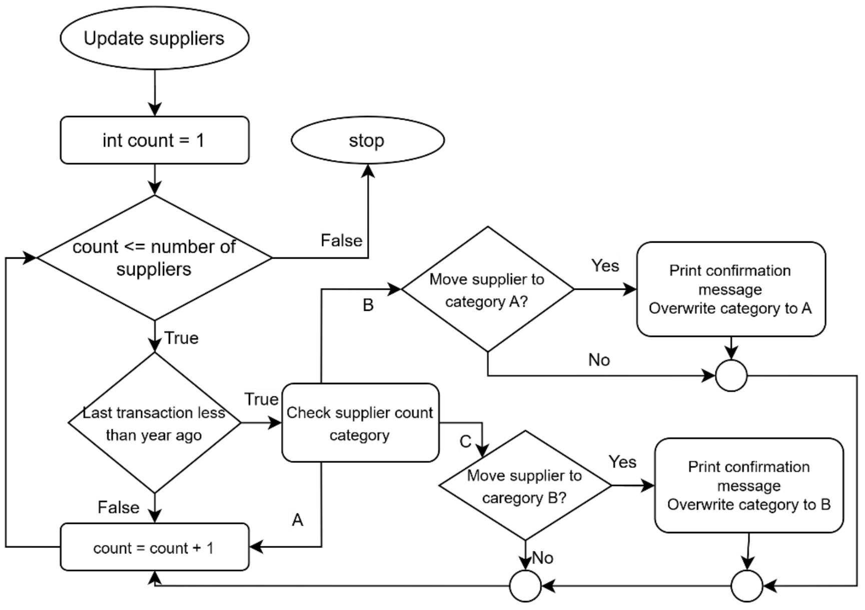
- The user should be able to:
    - View transaction in a clear and readable table, search by supplier name and category
    - View all the possible items for transactions
    - View a list of suppliers, filter them by category, and search by name
    - Input information (new transactions) by choosing from existing items and suppliers.
    - Add new items, suppliers, and transactions to the tables

- The program should:
    - update the tables automatically at each new entry
    - have a clear and simple interface
    - automatically fetch information that does not need to be typed manually
    - reject entries which are incorrect
    - run as a separate application

Criterion B: Design

Program's Working Plan

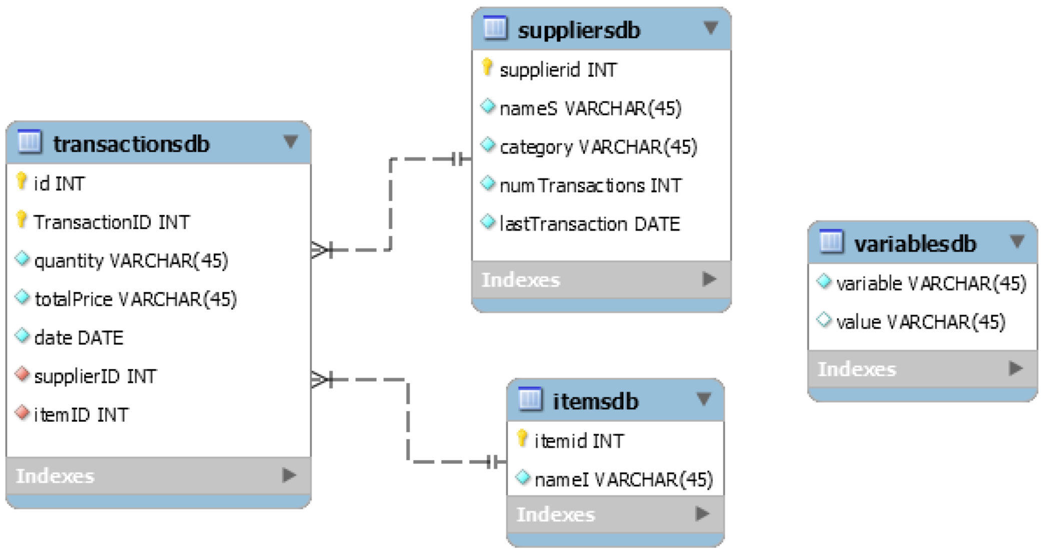

start
Add new transaction
No
Display transactions table
Edit transactions
No
stop
Yes
Yes
Choose supplier and product
Type in amount and price
Overwrite transaction details
Supplier has more than 6 transactions AND last transaction less than year ago
True
Automatically complete transaction details:
unique ID
supplier's category
Move supplier to category C?
Yes
Print confirmation message
Overwrite category to C
False
No
B
Check supplier's category
A
C
Update suppliers
Supplier has more than 3 transactions AND last transaction less than year ago
True
Move supplier to category B?
Yes
Print confirmation message
Overwrite category to B
False
No


Update suppliers
int count = 1
stop
count <= number of suppliers
False
Move supplier to category A?
Yes
Print confirmation message
Overwrite category to A
B
True
No
Last transaction less than year ago
True
Check supplier count category
C
Move supplier to carегory B?
Yes
Print confirmation message
Overwrite category to B
False
A
count = count + 1
No

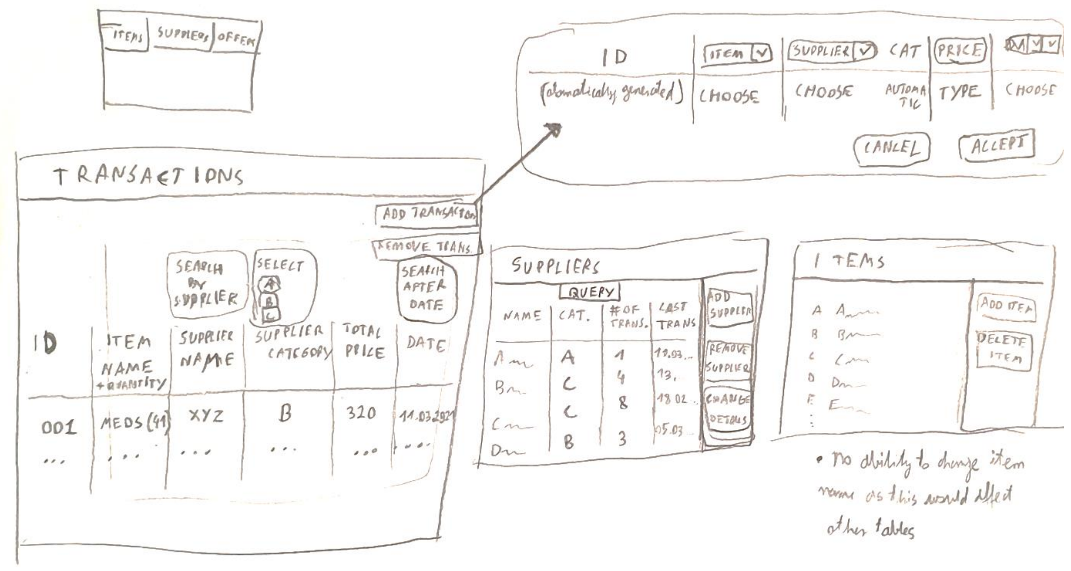
To make the work of the client more efficient all tasks should be automated if possible. The program will ask to transfer supplier to a higher category if they completed 3 transactions (see Appendix 1). To make sure the supplier's reliability hasn't changed since the last transaction, whether it was less than a year ago is also checked. If older than one year, the function **Update Suppliers** should transfer the related supplier one category below.

Database Design (see Appendix 2 for initial prototype)

My program will use a relational database with 3 linked tables and one separate for storing variables. Table **transactionsdb** has a many-to-one relationship with **suppliersdb** and **itemsdb** because many transactions may have the same supplier / item. Referencing is achieved through 2 additional variables (red) in transactions table.

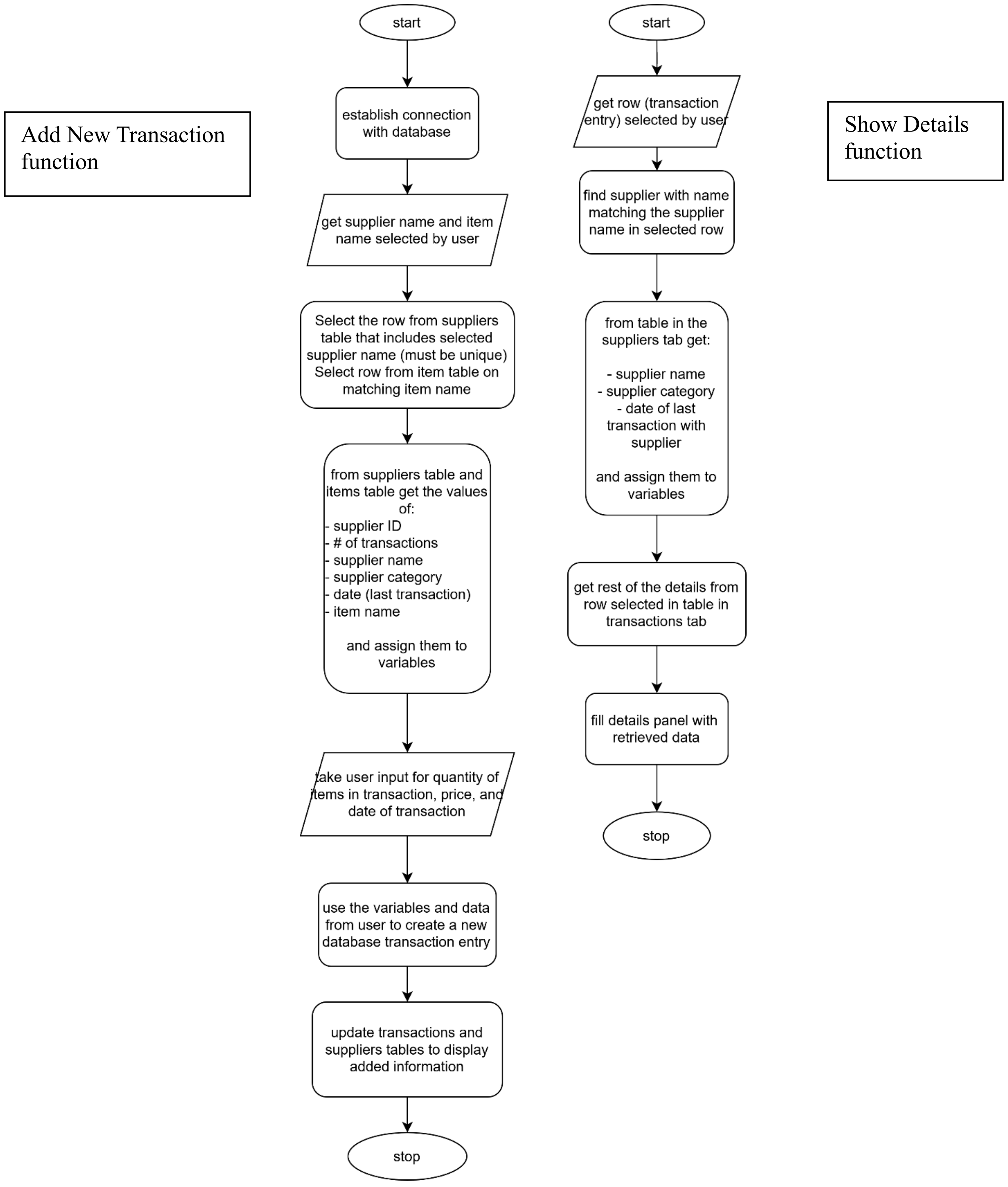


Graphical User Interface Design (showing the 3 tabs: transactions, suppliers, items)



Example functions: “Add New Transaction” and “Show Details”

Either function represents a different method of displaying data to the user that comes from 2 different database tables. While the first method fetches data directly from the database (which is slow), the second uses an already populated table from a different tab in the program.


Add New Transaction function
start
establish connection with database
get supplier name and item name selected by user
Select the row from suppliers table that includes selected supplier name (must be unique) Select row from item table on matching item name
from suppliers table and items table get the values of: - supplier ID - # of transactions - supplier name - supplier category - date (last transaction) - item name and assign them to variables
take user input for quantity of items in transaction, price, and date of transaction
use the variables and data from user to create a new database transaction entry
update transactions and suppliers tables to display added information
stop
Show Details function
start
get row (transaction entry) selected by user
find supplier with name matching the supplier name in selected row
from table in the suppliers tab get: - supplier name - supplier category - date of last transaction with supplier and assign them to variables
get rest of the details from row selected in table in transactions tab
fill details panel with retrieved data
stop

Collections and Objects of the Class TransacationsForm:

The whole program is included within just one class **TransactionsForm** which is called when the program starts. Below are functions that are called when different elements of the GUI are interacted with.

```
Connection sqlConn = null; // connection to the database
ResultSet rs = null; // table containing results of the current query
Statement stmt = null; // allows me to Query/insert/update/delete from and into the database
PreparedStatement pst = null; // allows me to interact with db (passing variables is possible)

public String[] supplierNames = new String[100];
// setting the limit of existing suppliers to 100
// this array will be used to display the drop-down menu
// for supplier names on the transactions panel

public String[] itemNames = new String[100];
// setting the limit of existing items to 100
// this array will be used to display the drop-down menu
// for item names on the transactions panel
```

First four objects will allow me to connect to and interact with the database. Arrays **supplierNames** and **itemNames** hold a maximum of 100 entries and will allow me to access the names from any point in the program.

Functions / Buttons (only the most relevant functions are presented)

Function Name	Purpose	Description
AddNewTransaction (Button)	Insert details provided by the user into the database	Input Required: supplier name, item name, quantity, price, date
AddNewSupplier (Button)		Input Required: supplier name, supplier category
AddNewItem (Button)		Input Required: item name
UpdateSuppliers	Maintain the correct category for each supplier	Checks number of supplier's transactions and whether last one older than 1 year.
ShowDetails	Present the user with a clear and readable details panel	Fills details panel with information from selected table row (when clicked upon)
Filter	Allows the user to see suppliers from selected category only	
RemoveTransaction (Button)	Delete information from database	Number of transactions of related supplier should decrease by 1
RemoveSupplier (Button)		All related transactions should be deleted too (warn the user)
RemoveItem (Button)		

Test plan table

Number of test	Action Test	Method of testing (expected result)
1	Change between the 3 tables	See if correct, easy to read tables are displayed with adequate rows and columns
2	Add new transaction / supplier / item	A new row should appear in the corresponding table containing all necessary details
3	Search transactions/suppliers by supplier name and category	Only supplier with the input name or category should appear in the transactions (or suppliers) table
4	Display details in the Details Panel	Correct transaction / supplier details should appear in the Details Panel when a row is selected
5	Edit details	New values can be written in the place of old ones and are then correctly displayed in the corresponding table
6	Automatic category change test (Supplier Update test)	After each 3 transactions, the program should prompt the user with information about the supplier and suggest an appropriate category shift.
7	Valid input test	New transactions should only be created when supplier and item are selected. New supplier and item should be created only if name is specified.
8	Data persistence test (program can run as a separate application)	Data should be the same after closing and re-opening the program

Criterion B: Record of tasks

Task number	Planned action	Planned outcome	Time estimated	Target completion date	Criterion
1	Deciding on a project	Coming up with a few ideas for my project.	1 week	24/03/2020	A
2	First discussion about my ideas with the teacher	Knowing what to avoid when searching for a client and designing a solution	1 day	30/03/2020	A
3	Searching for a possible client. First discussion with the client	multiple possible clients found and contacted.	3 weeks	21/04/2020	A
4	Deciding on the client and the type of my solution	An idea for a specific solution with one client in mind.	1 week	26/05/2020	A
5	Research technologies which I could use in my project and contact the teacher	appropriate programming language and tools to develop my application are found and discussed with the teacher	1 month	21/06/2020	A,B
6	Create a schedule with due dates	Tasks are assigned dates and put on schedule	1 week	25/06/2020	A
7	Define the scope of the project	Find out what must be implemented and what is out of scope.	1 week	15/07/2020	A
8	Summer holidays	Taking time to rest	2 months	-	-

9	State the criteria for success	The information gathered from interviews with the client and feedback from the teacher is used to make a list of crucial features for the solution	1 week	07/09/2020	B
10	Design a GUI prototype	Decided on how I want to lay out the input forms and display the transaction information tables	2 days	30/09/2020	A, B
11	Identify the classes needed for the program	A list of possible classes containing information about how they are useful to the program	1 week	03/10/2020	B
12	Write the code in Java using NetBeans	First fragments of code are written inside the IDE	1-2 months	14/11/2020	C
13	Develop a working prototype	The program has some basic functionality and responds to human input	1-2 months	10/01/2021	C
14	Consultation with the client and the teacher	I received feedback from the teacher and the client regarding my program	1 week	18/01/2021	B
15	Feedback is implemented into the program	Program has features which were requested by the client. It is also improved to include teacher's suggestions	2 weeks	31/01/2021	C, E

16	Complete the final version of the program	There is now a complete, working version of the program	2 weeks	10/02/2021	C
17	Make a video explaining the functionality of my product	The video is complete and edited	1 week	17/02/2021	D
18	Think of possible extensions of my project and evaluate the success criteria	The initial criteria are reflected upon and possible improvements are noted	1 week	02/03/2021	E

Criterion C: Development

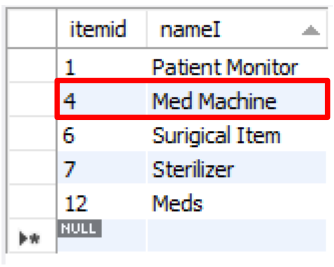
Techniques used to develop the solution:

- MySQL Database
- MySQL Workbench
- JDBC Connector
- GUI (Java Swing)
- Methods
- Conditions and switches
- Boolean flags

MySQL Database

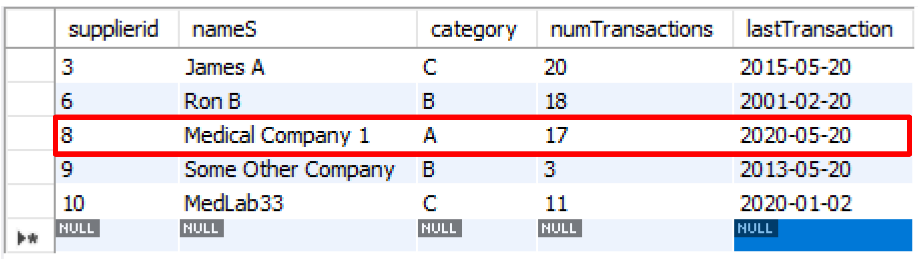
Storing data is achieved through a relational MySQL database on a localhost server. I used MySQL Workbench to create, manipulate, and view the database. "Select all" query returns the whole table:

```
SELECT * FROM programdatabase.itemsdb;
```



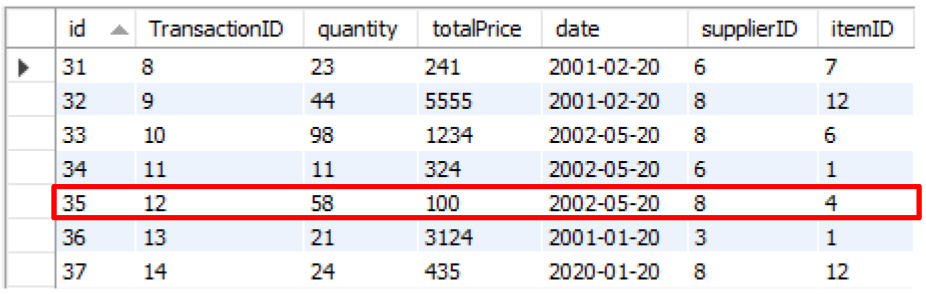
itemid	nameI
1	Patient Monitor
4	Med Machine
6	Surigical Item
7	Sterilizer
12	Meds
NULL	

```
SELECT * FROM programdatabase.suppliersdb;
```



supplierid	nameS	category	numTransactions	lastTransaction
3	James A	C	20	2015-05-20
6	Ron B	B	18	2001-02-20
8	Medical Company 1	A	17	2020-05-20
9	Some Other Company	B	3	2013-05-20
10	MedLab33	C	11	2020-01-02
NULL	NULL	NULL	NULL	NULL

```
SELECT * FROM programdatabase.transactionsdb;
```



id	TransactionID	quantity	totalPrice	date	supplierID	itemID
31	8	23	241	2001-02-20	6	7
32	9	44	5555	2001-02-20	8	12
33	10	98	1234	2002-05-20	8	6
34	11	11	324	2002-05-20	6	1
35	12	58	100	2002-05-20	8	4
36	13	21	3124	2001-01-20	3	1
37	14	24	435	2020-01-20	8	12

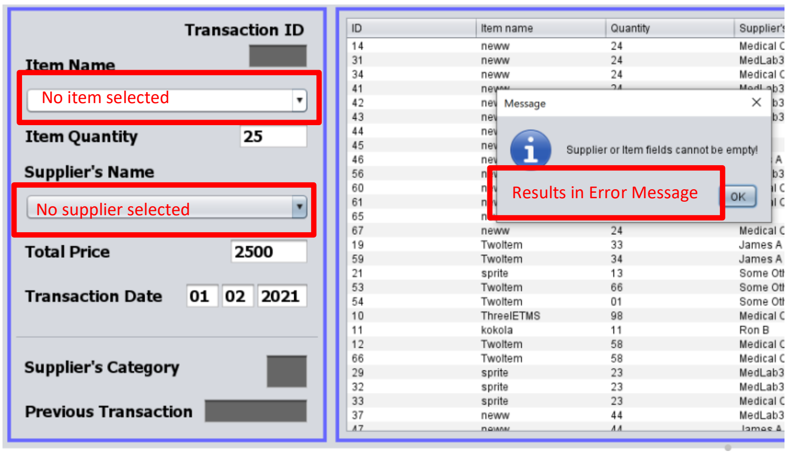
Relations are formed using a common key (**supplierid** for suppliers and **itemid** for items). The fields marked red are related. Transaction with **TransactionID** 12 includes 58 "Med Machines" and is made by supplier "Medical Company 1". I used MySQL Workbench to make the fields unique, hence there can be no two items / suppliers with the same id. Field **TransactionID** is also unique, which prevents conflict when identifying the correct transaction.

The 4th table includes variables. Their values can be fetched at any point in the program if the database is connected. This allows for great extensibility, as new persistent and globally accessible variables can be easily added. Currently, only one **numTrans** (number of total transactions) is stored – it is used to find the next non-conflicting **TransactionID** when creating a transaction.

```
SELECT * FROM programdatabase.variablesdb;
```

	variable	value
▶	numTrans	67
✱	NULL	NULL

MySQL Workbench allowed me to set all values to Non-Null. When the user attempts to enter a null value, the program will catch an SQL exception and message will be displayed:



Transaction ID

Item Name: No item selected

Item Quantity: 25

Supplier's Name: No supplier selected

Total Price: 2500

Transaction Date: 01 02 2021

Supplier's Category

Previous Transaction

ID	Item name	Quantity	Supplier's
14	neww	24	Medical C
31	neww	24	MedLab3
34	neww	24	Medical C
41	neww	24	MedLab3
42	nev		b3
43	nev		b3
44	nev		
45	nev		
46	nev		A
56	n		b3
60	n		l C
61	n		l C
65	n		
67	neww	24	Medical C
19	TwoItem	33	James A
59	TwoItem	34	James A
21	sprite	13	Some Ot
53	TwoItem	66	Some Ot
54	TwoItem	01	Some Ot
10	ThreeIETMS	98	Medical C
11	kokola	11	Ron B
12	TwoItem	58	Medical C
66	TwoItem	58	Medical C
29	sprite	23	MedLab3
32	sprite	23	MedLab3
33	sprite	23	Medical C
37	neww	44	MedLab3
47	neww	44	James A

Message

Supplier or Item fields cannot be empty!

JDBC Connector

Netbeans provides an easy and fast method of database-program communication with the use of JDBC Connector. At the start of **TransactionsForm** class I defined 3 Strings.

```
public class TransactionsForm extends javax.swing.JFrame {

    // connecting to the database on localhost
    private static final String username="root"; // the login is root at default
    private static final String password="DgJk34pop_"; // the password will be changed once the program is complete
    private static final String dataConn="jdbc:mysql://localhost:3306/programdatabase";
    // database is located at localhost (eventually client's  computer) on port 3306
```

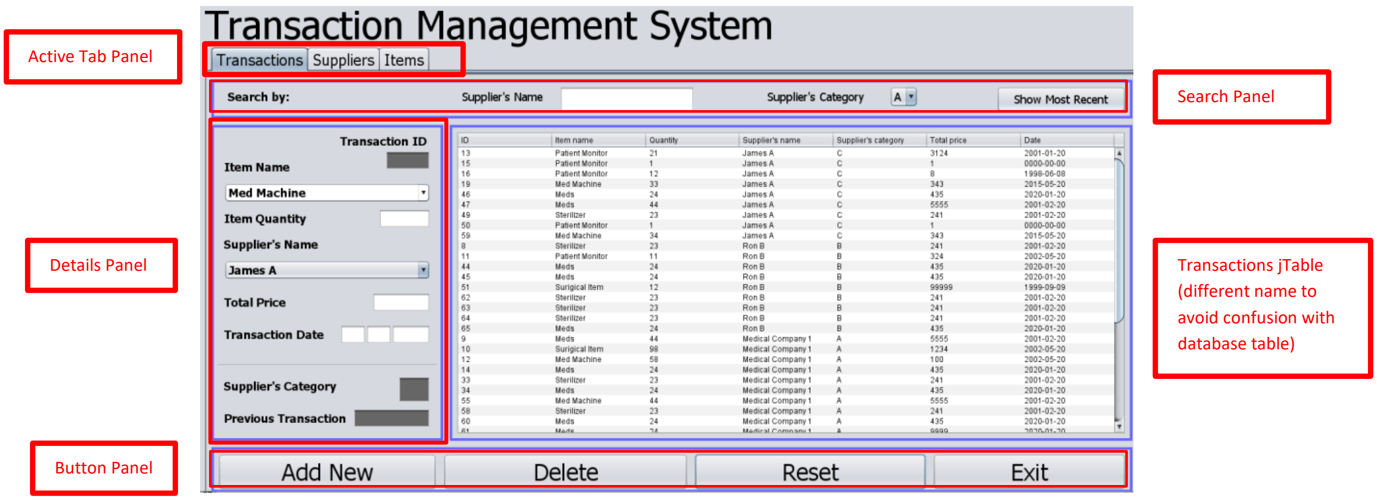
Each time the programs interacts with the database, the following code is run:

```
try
{
    Class.forName("com.mysql.cj.jdbc.Driver");
    sqlConn = DriverManager.getConnection(dataConn, username, password);

    /*
    /
    /   Some code to be executed...
    /
    */

} catch (Exception ex) {
    JOptionPane.showMessageDialog(null, ex);
}
```

Note that the connection attempt is surrounded in a try-catch clause. If the connection fails, an SQL Exception is caught.

GUI

I used Java Swing to create the GUI. I defined the elements (red) according to my initial design. The following screenshot shows the first loaded window (Transactions Tab).



Search by: Supplier's Name Supplier's Category A

Transaction ID

Item Name: Med Machine

Item Quantity

Supplier's Name: James A

Total Price

Transaction Date

Supplier's Category

Previous Transaction

ID	Item name	Quantity	Supplier's name	Supplier's category	Total price	Date
13	Patient Monitor	21	James A	C	3124	2001-01-20
15	Patient Monitor	1	James A	C	1	0000-00-00
16	Patient Monitor	12	James A	C	8	1998-06-08
19	Med Machine	33	James A	C	343	2015-05-20
46	Meds	24	James A	C	435	2020-01-20
47	Meds	44	James A	C	5555	2001-02-20
49	Sterilizer	23	James A	C	241	2001-02-20
50	Patient Monitor	1	James A	C	1	0000-00-00
59	Med Machine	34	James A	C	343	2015-05-20
8	Sterilizer	23	Ron B	B	241	2001-02-20
11	Patient Monitor	11	Ron B	B	324	2002-05-20
44	Meds	24	Ron B	B	435	2020-01-20
45	Meds	24	Ron B	B	435	2020-01-20
51	Surigical Item	12	Ron B	B	99999	1999-09-09
62	Sterilizer	23	Ron B	B	241	2001-02-20
63	Sterilizer	23	Ron B	B	241	2001-02-20
64	Sterilizer	23	Ron B	B	241	2001-02-20
65	Meds	24	Ron B	B	435	2020-01-20
9	Meds	44	Medical Company 1	A	5555	2001-02-20
10	Surigical Item	98	Medical Company 1	A	1234	2002-05-20
12	Med Machine	58	Medical Company 1	A	100	2002-05-20
14	Meds	24	Medical Company 1	A	435	2020-01-20
33	Sterilizer	23	Medical Company 1	A	241	2001-02-20
34	Meds	24	Medical Company 1	A	435	2020-01-20
55	Med Machine	44	Medical Company 1	A	5555	2001-02-20
58	Sterilizer	23	Medical Company 1	A	241	2001-02-20
60	Meds	24	Medical Company 1	A	435	2020-01-20
61	Meds	24	Medical Company 1	A	9999	2020-01-20

The Active Tab Panel allows to switch between the 3 tabs (Transactions, Suppliers, Items). The jTable displays all the information that the user needs to know about all transaction. The Details panel displays transaction-specific information and the date of previous transaction of related supplier. It also allows for editing and adding new transactions. Button Panel allows the user to add and delete transactions, reset the view and exit the application. Search panel lets the user sort and filter transactions. Screenshots of other 2 tabs are availiable in appendicies (see Appendix 3 & 4).

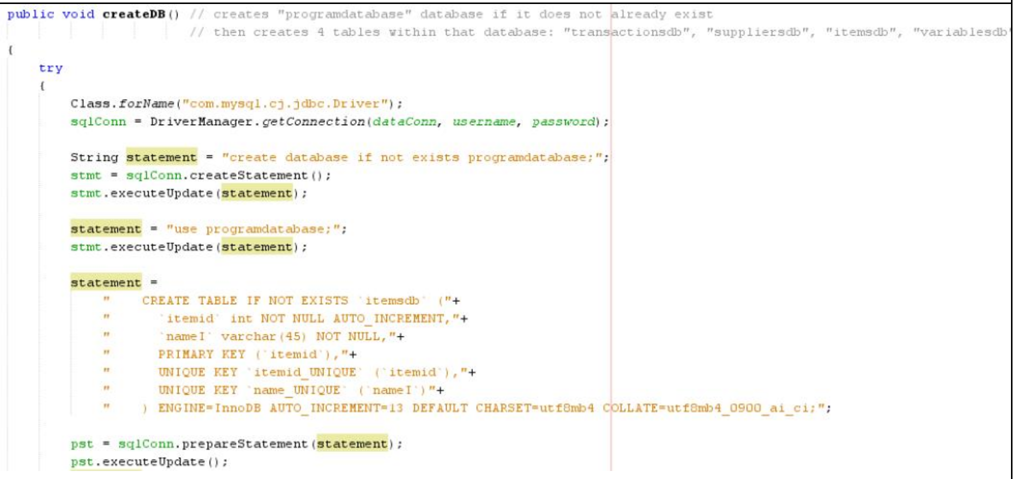
Methods / Functions

Before performing any operations, the program calls the following methods:

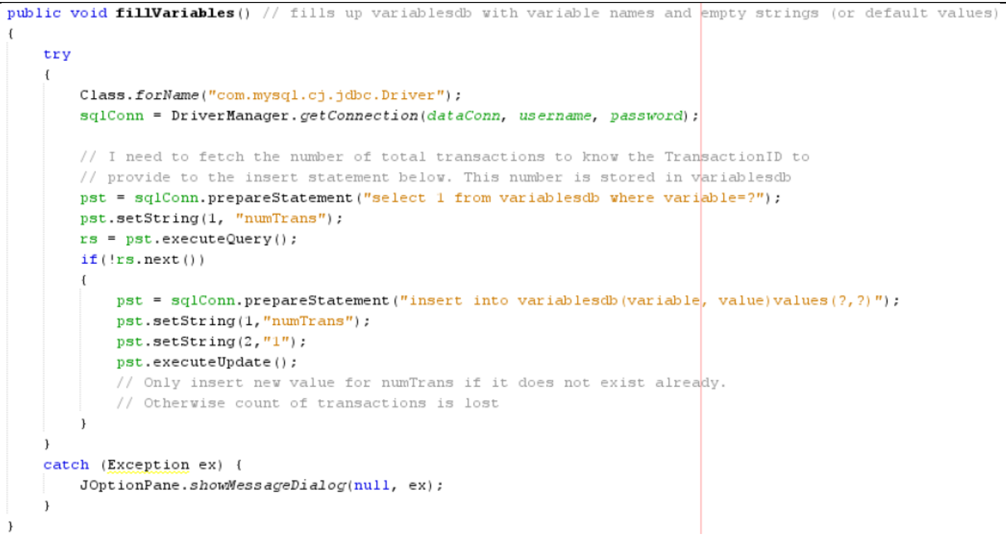
```
public TransactionsForm() {
    initComponents();
    createDB(); // creating database with 4 tables: transactions, suppliers, items and variables
    fillVariables(); // setting up variables to be stored in the db
    filterAll(); // for filtering transactions table
    updateDB(); // updating table view for transactions tab
    updateSuppliers(); //  updating table view for suppliers tab
    updateItems(); // updating table view for items tab
```

Functionality of those methods is described in the table below.

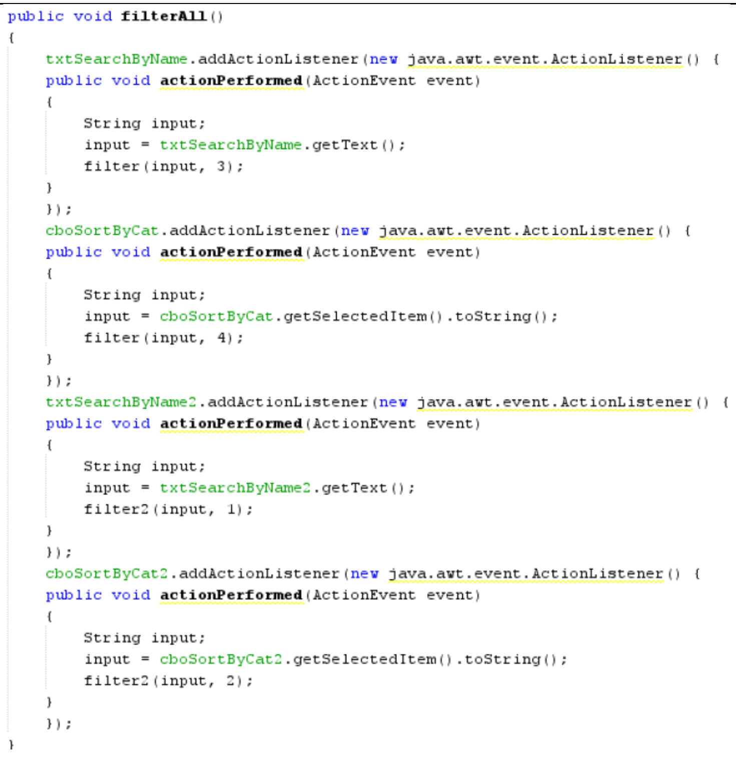
Function Name	createDB()

Purpose / Functionality	If the user is using the program for the first time, a localhost MySQL server (with password and login) needs to be set up. The rest is done automatically by the program. The creation statement for each table was reverse-engineered using MySQL Workbench.
Code	
Comment	The code continues to create 3 more tables in a similar way.

```
public void createDB() // creates "programdatabase" database if it does not already exist
                       // then creates 4 tables within that database: "transactionsdb", "suppliersdb", "itemsdb", "variablesdb"
{
    try
    {
        Class.forName("com.mysql.cj.jdbc.Driver");
        sqlConn = DriverManager.getConnection(dataConn, username, password);

        String statement = "create database if not exists programdatabase;";
        stmt = sqlConn.createStatement();
        stmt.executeUpdate(statement);

        statement = "use programdatabase;";
        stmt.executeUpdate(statement);

        statement =
            "    CREATE TABLE IF NOT EXISTS `itemsdb` ("+
            "      `itemid` int NOT NULL AUTO_INCREMENT,"+
            "      `nameI` varchar(45) NOT NULL,"+
            "     PRIMARY KEY (`itemid`),"+
            "     UNIQUE KEY `itemid_UNIQUE` (`itemid`),"+
            "     UNIQUE KEY `name_UNIQUE` (`nameI`)"+
            "    ) ENGINE=InnoDB AUTO_INCREMENT=13 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;";

        pst = sqlConn.prepareStatement(statement);
        pst.executeUpdate();
```

Function Name	fillVariables()
Purpose / Functionality	Prepares **variablesdb** table (in database) for storing **numTrans** variable with default value 1. Needed only when first using the program or when **numTrans** entry is accidentaly deleted (which should never happen).
Code	
Comment	Checks whether entry exists before inserting into database. Can be extended if more persistent (maintains value after restart) variables are added in the future.

```
public void fillVariables() // fills up variablesdb with variable names and empty strings (or default values)
{
    try
    {
        Class.forName("com.mysql.cj.jdbc.Driver");
        sqlConn = DriverManager.getConnection(dataConn, username, password);

        // I need to fetch the number of total transactions to know the TransactionID to
        // provide to the insert statement below. This number is stored in variablesdb
        pst = sqlConn.prepareStatement("select 1 from variablesdb where variable=?");
        pst.setString(1, "numTrans");
        rs = pst.executeQuery();
        if(!rs.next())
        {
            pst = sqlConn.prepareStatement("insert into variablesdb(variable, value)values(?,?)");
            pst.setString(1,"numTrans");
            pst.setString(2,"1");
            pst.executeUpdate();
            // Only insert new value for numTrans if it does not exist already.
            // Otherwise count of transactions is lost
        }
    }
    catch (Exception ex) {
        JOptionPane.showMessageDialog(null, ex);
    }
}
```

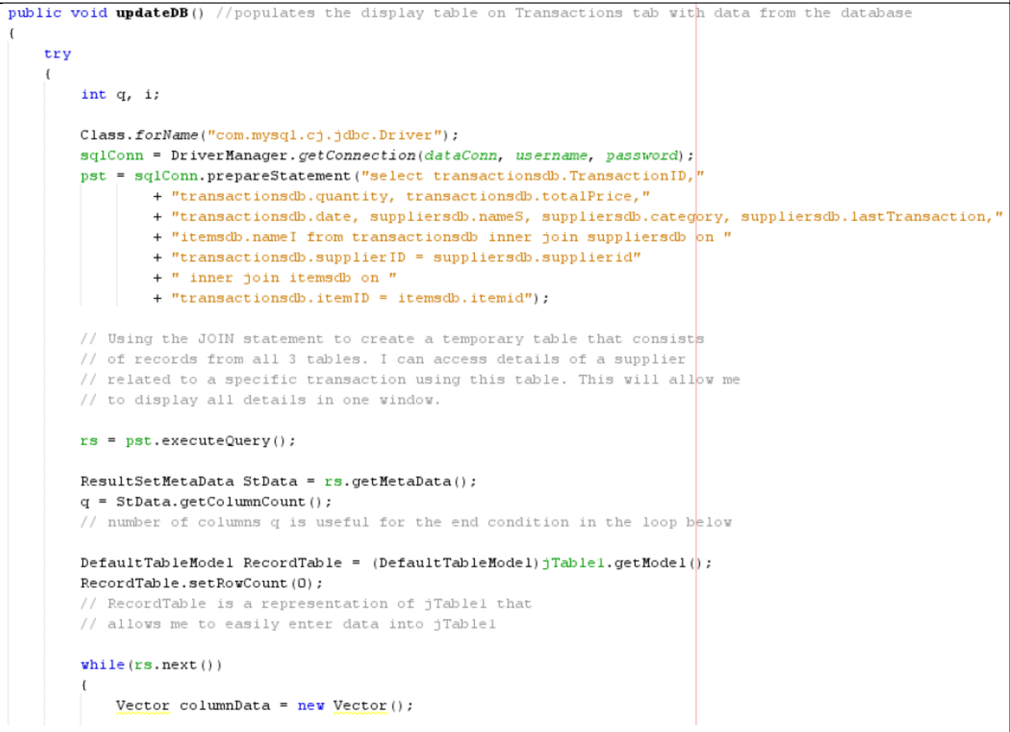
Function Name	filterAll()
Purpose / Functionality	Sets up ActionListeners which allow filtering (by name and by category) in transactions and suppliers tables (hence 2*2 code blocks).
Code	

```
public void filterAll()
{
    txtSearchByName.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(ActionEvent event)
    {
        String input;
        input = txtSearchByName.getText();
        filter(input, 3);
    }
    });
    cboSortByCat.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(ActionEvent event)
    {
        String input;
        input = cboSortByCat.getSelectedItem().toString();
        filter(input, 4);
    }
    });
    txtSearchByName2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(ActionEvent event)
    {
        String input;
        input = txtSearchByName2.getText();
        filter2(input, 1);
    }
    });
    cboSortByCat2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(ActionEvent event)
    {
        String input;
        input = cboSortByCat2.getSelectedItem().toString();
        filter2(input, 2);
    }
    });
}
```

Comment	Accessible through Search Panel in the GUI. Refers to functions filter() and filter2() which use the built-in jTable sorter functionality to filter given a String.

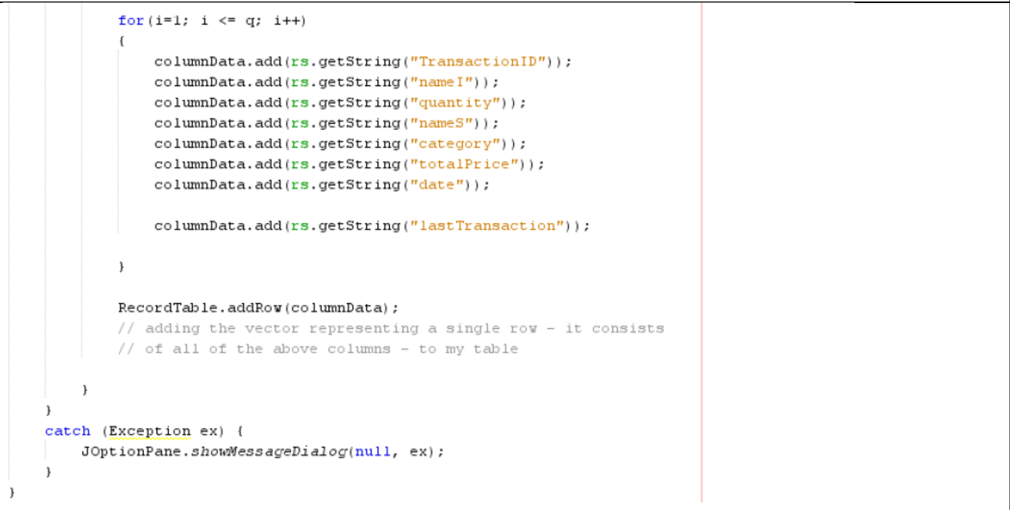
Function Name	updateDB()
Purpose / Functionality	Sets up the table model for viewing the jTable displayed in the transactions tab of the GUI. Data is fetched from **transactionsdb, suppliersdb, itemsdb** tables and inserted into the jTable in the view.

Code



```
public void updateDB() //populates the display table on Transactions tab with data from the database
{
    try
    {
        int q, i;

        Class.forName("com.mysql.cj.jdbc.Driver");
        sqlConn = DriverManager.getConnection(dataConn, username, password);
        pst = sqlConn.prepareStatement("select transactionsdb.TransactionID,"
                + "transactionsdb.quantity, transactionsdb.totalPrice,"
                + "transactionsdb.date, suppliersdb.nameS, suppliersdb.category, suppliersdb.lastTransaction,"
                + "itemsdb.nameI from transactionsdb inner join suppliersdb on "
                + "transactionsdb.supplierID = suppliersdb.supplierid"
                + " inner join itemsdb on "
                + "transactionsdb.itemID = itemsdb.itemid");

        // Using the JOIN statement to create a temporary table that consists
        // of records from all 3 tables. I can access details of a supplier
        // related to a specific transaction using this table. This will allow me
        // to display all details in one window.

        rs = pst.executeQuery();

        ResultSetMetaData StData = rs.getMetaData();
        q = StData.getColumnCount();
        // number of columns q is useful for the end condition in the loop below

        DefaultTableModel RecordTable = (DefaultTableModel)jTable1.getModel();
        RecordTable.setRowCount(0);
        // RecordTable is a representation of jTable1 that
        // allows me to easily enter data into jTable1

        while(rs.next())
        {
            Vector columnData = new Vector();
```

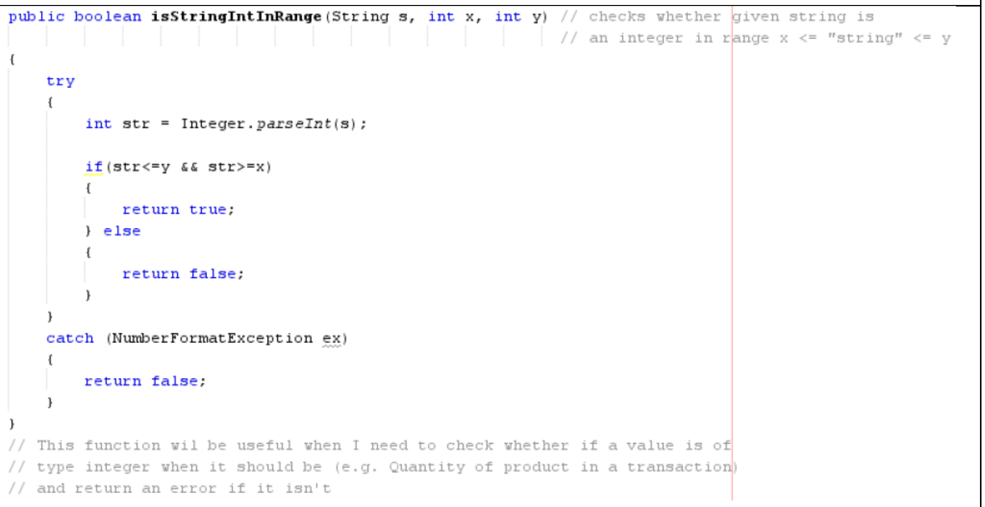
Code continued...



```
            for(i=1; i <= q; i++)
            {
                columnData.add(rs.getString("TransactionID"));
                columnData.add(rs.getString("nameI"));
                columnData.add(rs.getString("quantity"));
                columnData.add(rs.getString("nameS"));
                columnData.add(rs.getString("category"));
                columnData.add(rs.getString("totalPrice"));
                columnData.add(rs.getString("date"));

                columnData.add(rs.getString("lastTransaction"));

            }

            RecordTable.addRow(columnData);
            // adding the vector representing a single row - it consists
            // of all of the above columns - to my table

        }
    }
    catch (Exception ex) {
        JOptionPane.showMessageDialog(null, ex);
    }
}
```

Comment	Note that the result set of the SELECT query is a temporary table that includes the data that needs to be displayed in the transactions jTable. Similar functions exist for updating suppliers and items jTables. They are called each time the data is changed

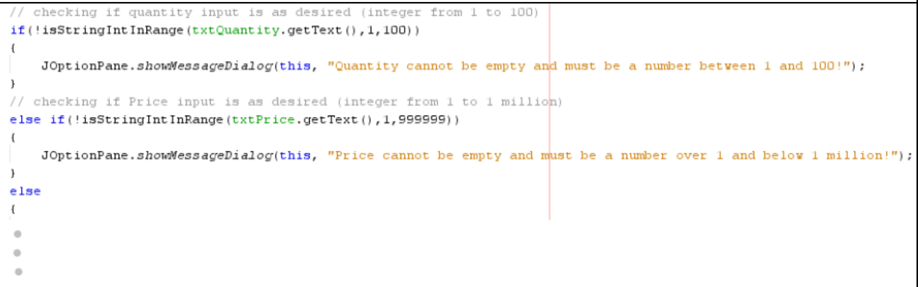
There is one more method that appears in the code but is not called at the start of the program:

Function Name	isStringIntInRange(String s, int x, int y)
Purpose / Functionality	Checks if the provided String can be parsed into Integer. If parse is successful it checks whether the integer is between number x and y inclusive. Returns boolean. It is used to validate user input from two text fields: **quantity** and **totalPrice.**
Code	

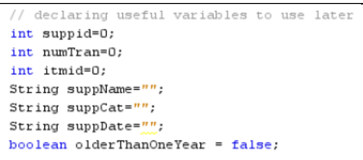
```
public boolean isStringIntInRange(String s, int x, int y) // checks whether given string is
                                                           // an integer in range x <= "string" <= y
{
    try
    {
        int str = Integer.parseInt(s);

        if(str<=y && str>=x)
        {
            return true;
        } else
        {
            return false;
        }
    }
    catch (NumberFormatException ex)
    {
        return false;
    }
}
// This function wil be useful when I need to check whether if a value is of
// type integer when it should be (e.g. Quantity of product in a transaction)
// and return an error if it isn't
```

Comment	This function can be extended and used in many other verification or validation forms

Each button from the GUI calls a purposeful function. The most complex case is **btnAddNewActionPerformed().** Fragments of code are shown in table below:

Function Name	btnAddNewActionPerformed()
Purpose / Functionality	Creates new transaction record from data inputted in the GUI Details Panel and data fetched from **suppliersdb** and **itemsdb** tables.
Checking if user input is as desired. Proceed if it is uses Boolean flag isStringIntInRange()	

```
// checking if quantity input is as desired (integer from 1 to 100)
if(!isStringIntInRange(txtQuantity.getText(),1,100))
{
    JOptionPane.showMessageDialog(this, "Quantity cannot be empty and must be a number between 1 and 100!");
}
// checking if Price input is as desired (integer from 1 to 1 million)
else if(!isStringIntInRange(txtPrice.getText(),1,999999))
{
    JOptionPane.showMessageDialog(this, "Price cannot be empty and must be a number over 1 and below 1 million!");
}
else
{
 •
 •
 •
```

Declaring useful variables	

```
// declaring useful variables to use later
int suppid=0;
int numTran=0;
int itmid=0;
String suppName="";
String suppCat="";
String suppDate="";
boolean olderThanOneYear = false;
```

Using primary and foreign keys (from **transactionsdb** and **suppliersdb** respectively) to get details of selected supplier.

Note the Boolean flag olderThanOneYear which will prevent supplier from advancing to higher category if true.

```java
// I need to find the supplier ID corresponding to the new transaction
// The rest of supplier details are updated from Event on combo box SupplierNames

pst = sqlConn.prepareStatement("select * from suppliersdb where nameS = ?");
pst.setString(1, cboSupplierName.getSelectedItem().toString());
rs = pst.executeQuery();
if(rs.next())
{
    suppName = rs.getString("nameS");
    suppCat = rs.getString("category");
    suppid = rs.getInt("supplierid");
    numTran = rs.getInt("numTransactions"); // useful when inserting into suppliersdb
    suppDate = rs.getDate("lastTransaction").toString();

    // Calculating the date difference (date now - date of transaction)
    final SimpleDateFormat df = new SimpleDateFormat( "yyyy-MM-dd" );
    try
    {
        final Date dateTrans = df.parse( suppDate ); // conversion from String to date
        final Date dateNow = new Date(); // getting current system date
        long timeDiff = dateNow.getTime()-dateTrans.getTime(); // calculating time difference

        // checking if the time difference is more than one year. if true, boolean becomes true.
        if((timeDiff/365*24*60*60*1000l)>=1)
        {
            olderThanOneYear = true;
        }

    } catch (ParseException ex) {
        Logger.getLogger(TransactionsForm.class.getName()).log(Level.SEVERE, null, ex);
    }

}
```

Same operation as above but for item.

```java
// I need to find the Item ID related to the item name in new transaction
pst = sqlConn.prepareStatement("select * from itemsdb where nameI = ?");
pst.setString(1, cboItemName.getSelectedItem().toString());
rs = pst.executeQuery();

if(rs.next())
{
    itmid = rs.getInt("itemid");
}
```

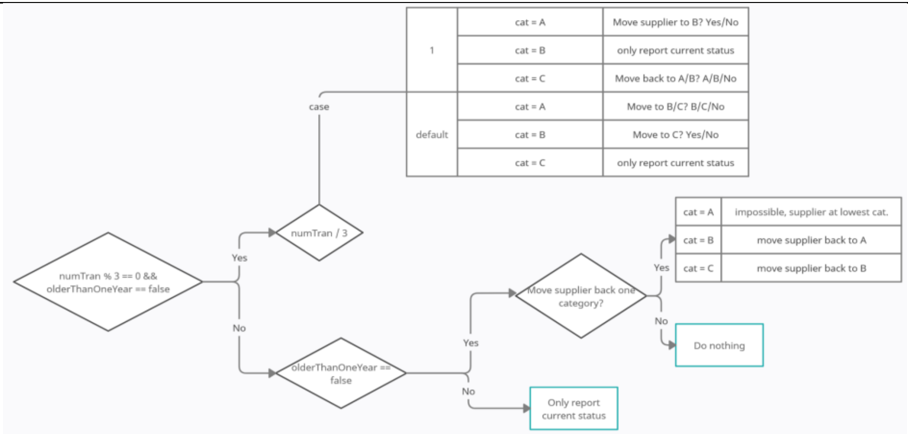
Updating number of total transactions (stored in database) by 1

```java
// fetch numTrans to insert into the table (how many transactions are there already)
int numTrans = 1;
pst = sqlConn.prepareStatement("select * from variablesdb where variable = ?");
pst.setString(1, "numTrans");
rs = pst.executeQuery();
while(rs.next())
{
    numTrans = rs.getInt("value");
    numTrans++;          // I need to update the value of numTrans to be increased by 1
                         // there is now one more transaction to the total count
}
```

Finally, inserting the fetched details (and user input) as a new transaction record into the database

```java
// Inserting complete data into Transactions table:
statement = "insert into transactionsdb(TransactionID,itemID,quantity,supplierID,"
    + " totalPrice,date)values(?,?,?,?,?,?)";
pst = sqlConn.prepareStatement(statement);

pst.setInt(1, numTrans);
pst.setInt(2, itmid);
pst.setString(3, (txtQuantity.getText()));
pst.setInt(4, suppid);
pst.setString(5, txtPrice.getText());
pst.setString(6, (String) txtDateY.getText()+"."+txtDateM.getText()+"."+txtDateD.getText());

pst.executeUpdate();
JOptionPane.showMessageDialog(this, "Transaction record added");
updateDB();
```

Program will suggest category shifts depending on number of supplier transactions, current category and date of last transaction (See decision tree diagram)

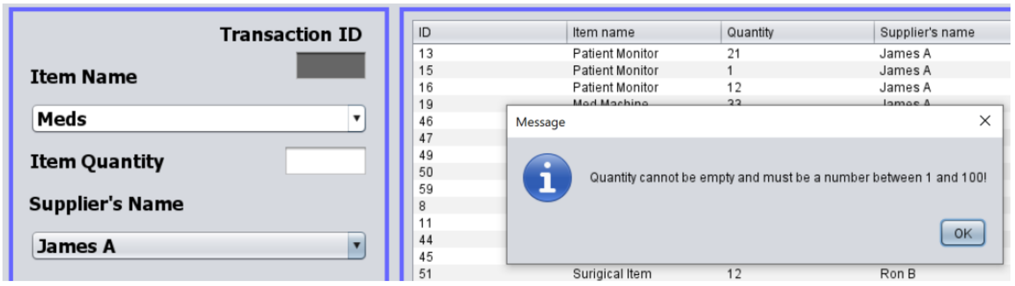


numTran % 3 == 0 && olderThanOneYear == false
Yes
numTran / 3
case
1
cat = A
Move supplier to B? Yes/No
cat = B
only report current status
cat = C
Move back to A/B? A/B/No
default
cat = A
Move to B/C? B/C/No
cat = B
Move to C? Yes/No
cat = C
only report current status
No
olderThanOneYear == false
Yes
Move supplier back one category?
Yes
cat = A
impossible, supplier at lowest cat.
cat = B
move supplier back to A
cat = C
move supplier back to B
No
Do nothing
No
Only report current status

Message Dialogs for Invalid Inputs / Operations (Error handling)

All of the messages below appear when user attempts to add new records

1) Invalid Quantity Error Message (allowed value: 1 <= input <= 100)



Transaction ID	
Item Name	Meds
Item Quantity	
Supplier's Name	James A

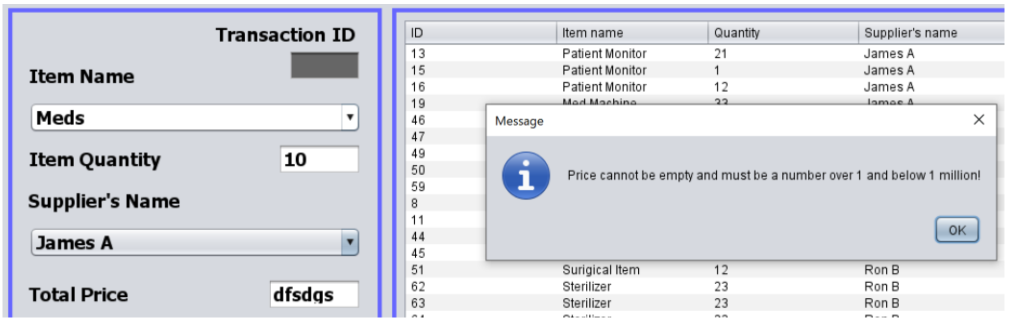
Message

Quantity cannot be empty and must be a number between 1 and 100!

OK

ID	Item name	Quantity	Supplier's name
13	Patient Monitor	21	James A
15	Patient Monitor	1	James A
16	Patient Monitor	12	James A
19	Med Machine	33	James A
46			
47			
49			
50			
59			
8			
11			
44			
45			
51	Surigical Item	12	Ron B

2) Invalid Price Error Message (allowed value: 1 <= input <= 1 000 000)



Transaction ID	
Item Name	Meds
Item Quantity	10
Supplier's Name	James A
Total Price	dfsdgs

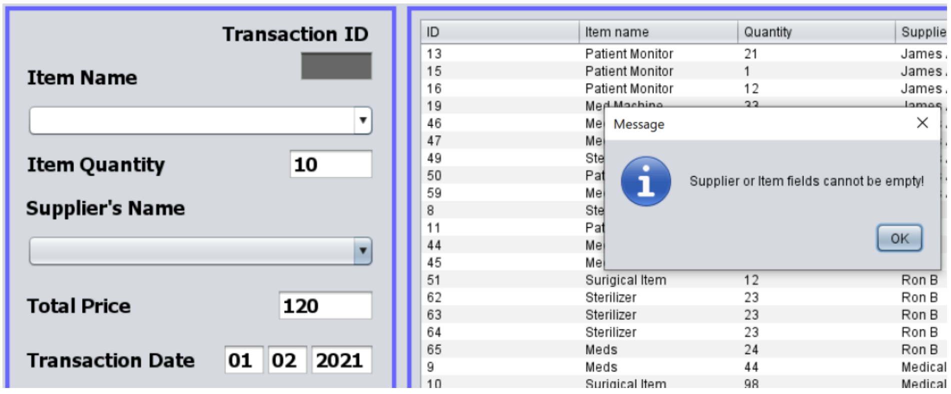
Message

Price cannot be empty and must be a number over 1 and below 1 million!

OK

ID	Item name	Quantity	Supplier's name
13	Patient Monitor	21	James A
15	Patient Monitor	1	James A
16	Patient Monitor	12	James A
19	Med Machine	33	James A
46			
47			
49			
50			
59			
8			
11			
44			
45			
51	Surigical Item	12	Ron B
62	Sterilizer	23	Ron B
63	Sterilizer	23	Ron B

3) Empty Supplier/Item Name Error Message



Transaction ID	
Item Name	
Item Quantity	10
Supplier's Name	
Total Price	120
Transaction Date	01 02 2021

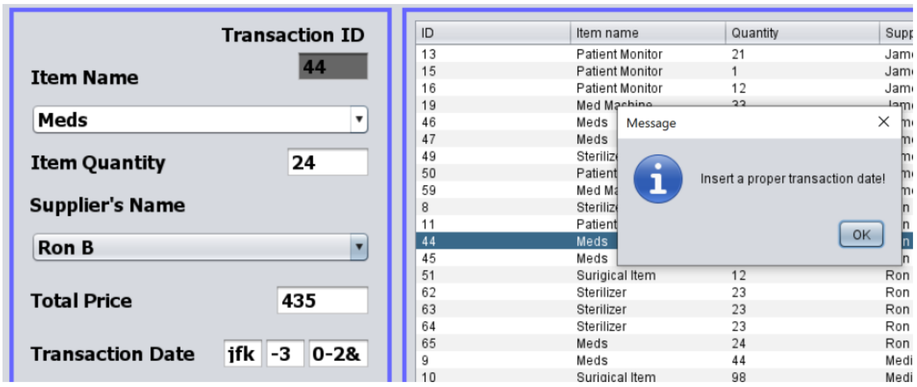
Message

Supplier or Item fields cannot be empty!

OK

ID	Item name	Quantity	Supplie
13	Patient Monitor	21	James
15	Patient Monitor	1	James
16	Patient Monitor	12	James
19	Med Machine	33	James
46	Me		
47	Me		
49	Ste		
50	Pat		
59	Me		
8	Ste		
11	Pat		
44	Me		
45	Me		
51	Surigical Item	12	Ron B
62	Sterilizer	23	Ron B
63	Sterilizer	23	Ron B
64	Sterilizer	23	Ron B
65	Meds	24	Ron B
9	Meds	44	Medical
10	Surigical Item	98	Medical

4) Invalid Transaction Date Error Message



Transaction ID	44
Item Name	Meds
Item Quantity	24
Supplier's Name	Ron B
Total Price	435
Transaction Date	ifk -3 0-2&

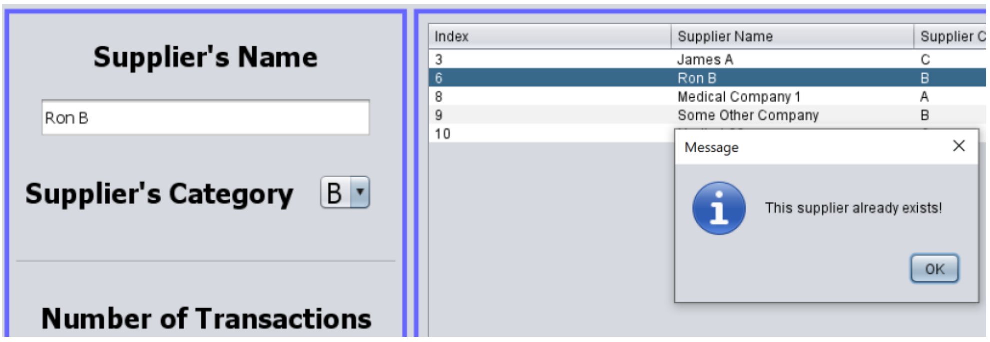
Message

Insert a proper transaction date!

OK

ID	Item name	Quantity	Supp
13	Patient Monitor	21	Jam
15	Patient Monitor	1	Jam
16	Patient Monitor	12	Jam
19	Med Machine	33	Jam
46	Meds		
47	Meds		
49	Sterilize		
50	Patient		
59	Med Ma		
8	Sterilize		
11	Patient		
44	Meds		
45	Meds		
51	Surigical Item	12	Ron
62	Sterilizer	23	Ron
63	Sterilizer	23	Ron
64	Sterilizer	23	Ron
65	Meds	24	Ron
9	Meds	44	Medi
10	Surigical Item	98	Medi

5) Supplier Name Conflict Message



Supplier's Name

Ron B

Supplier's Category B

Number of Transactions

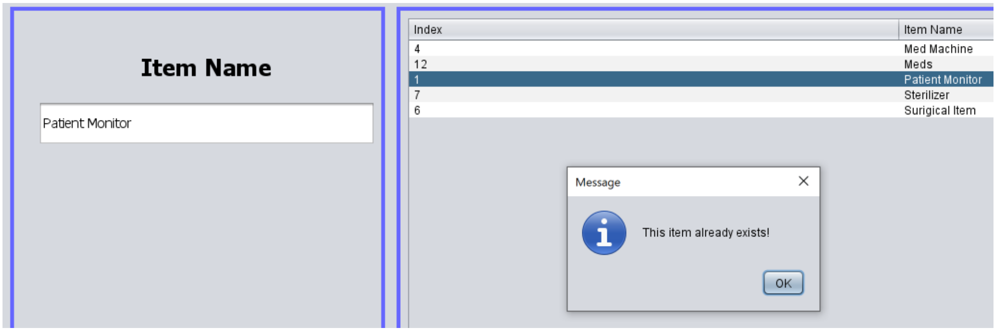
Index	Supplier Name	Supplier C
3	James A	C
6	Ron B	B
8	Medical Company 1	A
9	Some Other Company	B
10		

Message

This supplier already exists!

OK

6) Item Name Conflict Message



Item Name

Patient Monitor

Index	Item Name
4	Med Machine
12	Meds
1	Patient Monitor
7	Sterilizer
6	Surigical Item

Message

This item already exists!

OK

Word Count: 967

Criterion E: Evaluation

Meeting Success Criteria:

Criterion Number	Description	Criterion Met
1	The user has access to a simple graphical interface that allows them to view transactions in a clear and readable table (as well as Details Panel). Switching between tabs is intuitive and there are not too many confusing buttons / panels	✓
2	The user can search for transactions by providing the supplier's name or filter transactions by supplier's category	✓
3	The user can view, add, and delete all possible items for transactions	✓
4	The user can view a list of all suppliers and filter them by category or search for specific names. They can also add and delete suppliers.	✓
5	The user can input information about new transaction records by choosing from existing items and suppliers. (Rest of the details such as the price need to be typed manually)	✓
6	The program automatically updates the tables at each new entry (through update functions)	✓
7	Suggestions about moving suppliers between categories are provided to the user automatically, based on specific conditions. The user always has a choice as to whether to move the supplier or not	✓
8	Information that is not typed in (such as the supplier's category when adding a transaction record) is automatically fetched and displayed in the Details Panel	✓
9	The program runs as a separate application and saves data between sessions	✓

All of the criteria initially specified by the client have been met. The feedback (see Appendix 5) suggests that the program works well. The only issue was with last transaction dates as they did not update with each transaction. This was fixed in the current state of the program.

Recommendations for Further Development

Ideally, the program should connect with the internal database of the business so that the user would not have to type in all the data manually. To further increase the accessibility of information, transactions with suppliers of each category could be color-coded. As requested by the client (see Appendix 5), a login form could be implemented to increase the security of the application.

The client suggested that the program should indicate the progress of each transaction. This could be achieved by implementing another field to the transactions table. The value of that field would be represented by a progress bar in the GUI. The transaction progress could be increased from the Details Panel.

Extrapolating from the Client's feedback I believe that it would be beneficial to include one more tab with a timeline of transactions. This would help the client orient in the time differences between transactions and provide a more readable, visual representation of transaction dates.

Word Count: 442

Appendix

Appendix #1. Client Interview Transcript

03.11.2020

The conversation takes place through a phone call and is initialized by my father who is an employee of the small business. The transcript is formatted to include only the most relevant fragments.

Me: What are Your tasks as part of the trading department?
Client: I manage the transactions we make with our suppliers. We currently keep track of the transactions in paper documents. It happens that we take many offers from one supplier when we see that this supplier is reliable.

Me: Do you keep a list of all suppliers?

Client: We don't keep a list although there are documents to keep track of the transactions. Each time I am not sure about the details I simply check it in the papers. I am used to it working in such a way although having [the details] all in one place would definitely help save some time. I often have to remember which providers we already took offers from so that we can decide on what terms we have the transaction.

Me: What would you say is the main issue right now? Is there anything that could be improved or automated?

Client: As I said, although I am used to the current system, it still takes some time to find all the relevant details. An easily accessible table of the transactions through the computer would make a big difference in the management process. I think I would have to get used to the new system but it should be faster in the long run.

Me: Do you require to access the information outside of Your workspace? This is important because the program could work either online or offline. Does anybody else need access to the program?

Client: No, All of the tasks I perform are within the building I work at, on a windows computer. There is no need for an online program, especially that it would deal with information that is only relevant for myself.

Me: My idea is to develop an offline program to view and manage the transactions data, however, I would need to know more about the transactions themselves. Could you tell me if there are any distinct features of the suppliers? What kind of details do you need to remember about the suppliers?

Client: They are usually divided into 3 Categories. If we only ever did one transaction with that supplier they are in category A. As we make more transactions with that supplier they go into the category B and so on. It depends on the number of transactions and how big the transaction is. For example, after 3 transactions that supplier goes into the next category. When I need to access specific information about a transaction I usually check the kind of the product, how many previous transactions there were and whether there were any issues with the supplier. It might not be on time or not the full quota. Then we simply don't move that supplier into the next category.

Me: Do the suppliers often change? If so, I think the program should allow to add new entries for the suppliers.

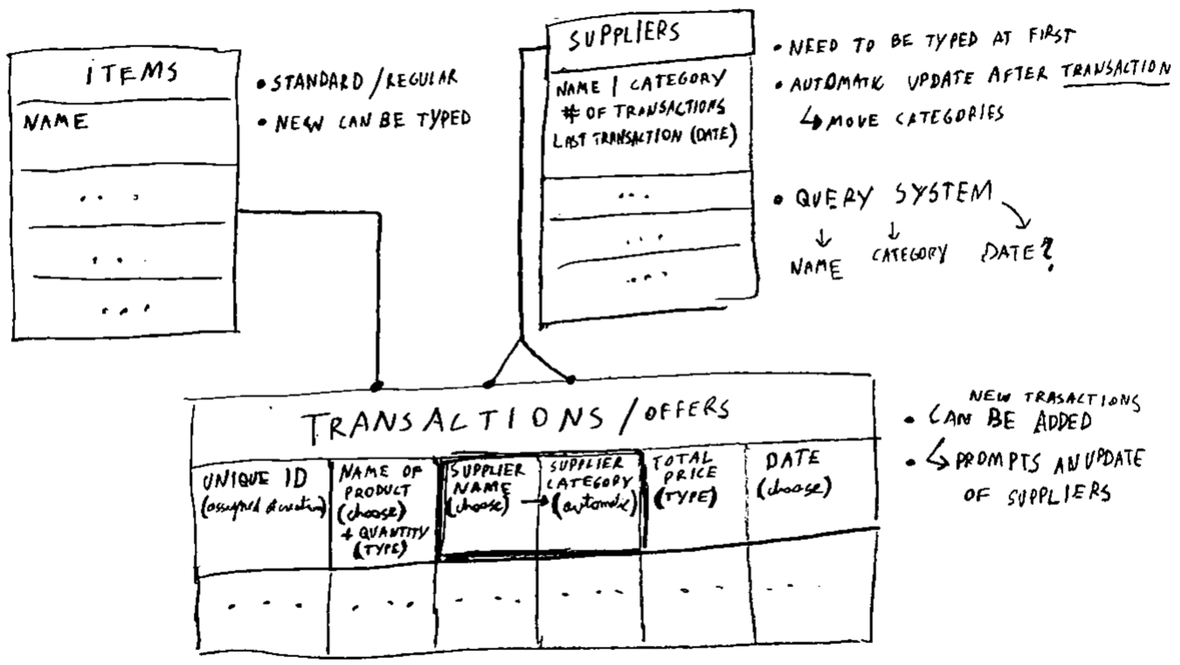
Client: We do have new suppliers from time to time. The equipment may also vary but mainly remains the same.

Me: I believe that there need to be separate lists of information for the medical equipment, the suppliers and all the transactions. This could be implemented using a database system. It would be accessible locally through an application on Your computer. Would this solution aid You in Your work?

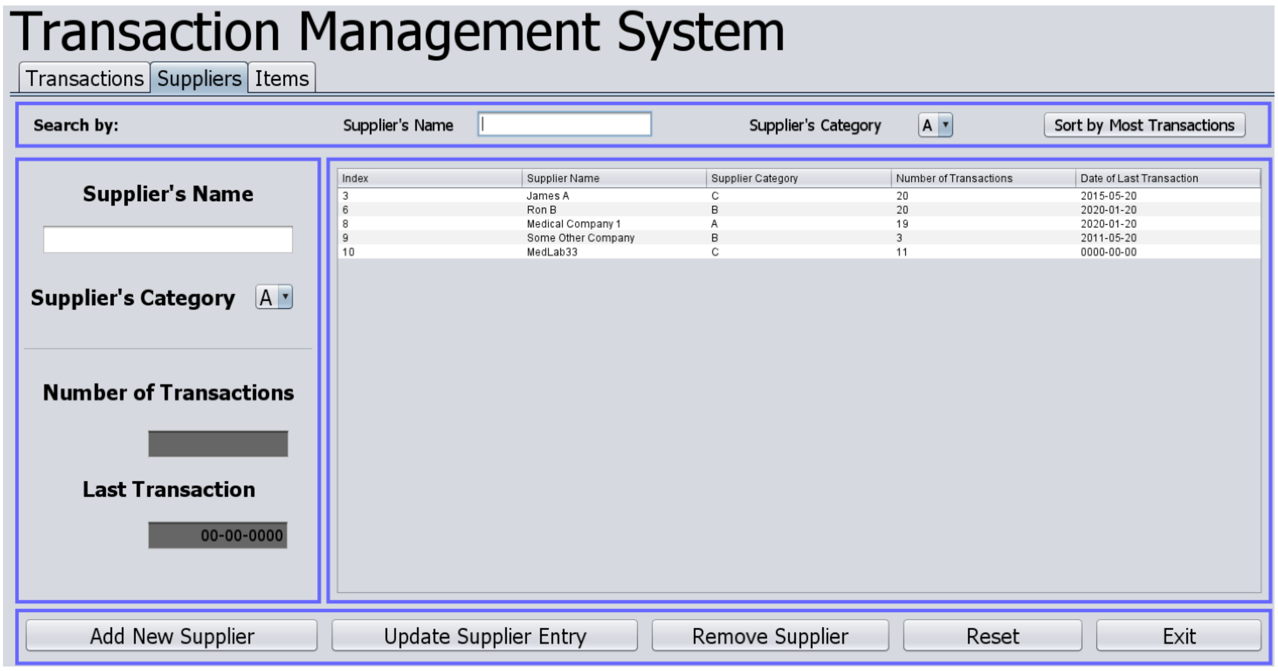
Client: Yes, I believe that the program would speed up my work. It would be very useful to access the information from one organised place on my computer.

Me: I will soon start working on the design. Thank you.

Appendix #2. Initial Database Design

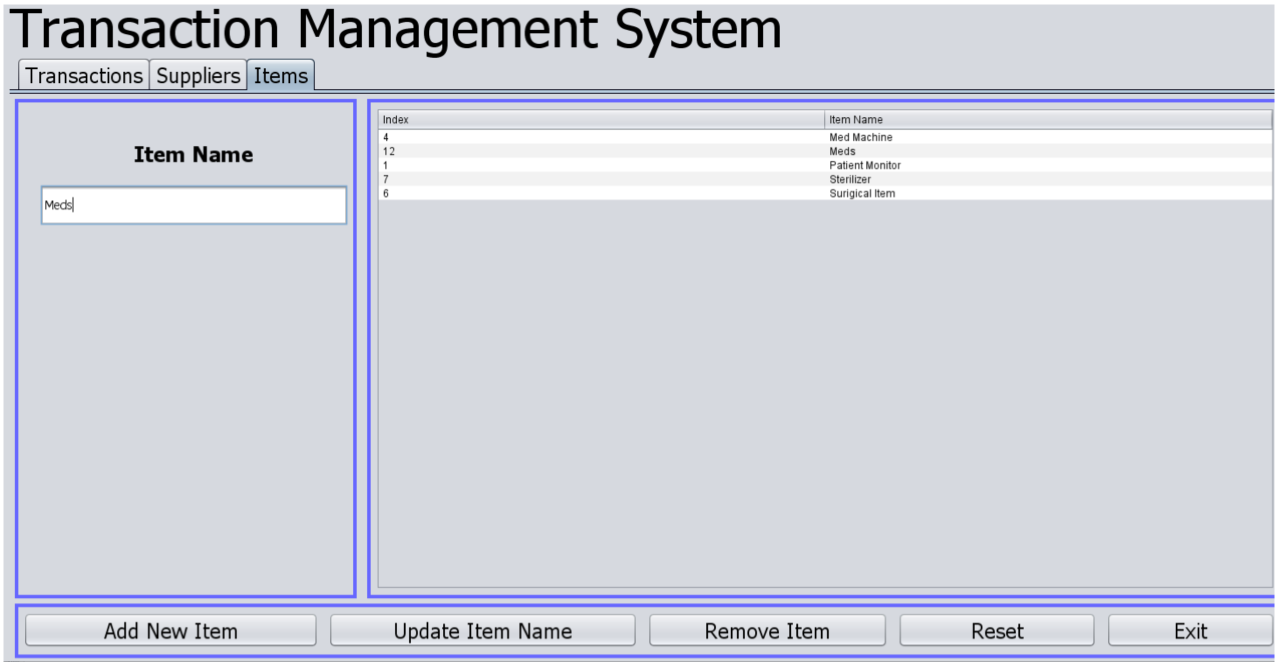


Appendix #3. GUI - Suppliers Tab



Index	Supplier Name	Supplier Category	Number of Transactions	Date of Last Transaction
3	James A	C	20	2015-05-20
6	Ron B	B	20	2020-01-20
8	Medical Company 1	A	19	2020-01-20
9	Some Other Company	B	3	2011-05-20
10	MedLab33	C	11	0000-00-00

Appendix #4. GUI - Items Tab



Item Name: Meds

Index	Item Name
4	Med Machine
12	Meds
1	Patient Monitor
7	Sterilizer
6	Surigical Item

Appendix #5. Client Feedback

Fragment of an E-mail from the client containing feedback regarding the application (translated from native language and reformatted):

18.01.2021

The application works well, I haven't encountered any issues when entering transactions expect for wrong dates appearing in the "Last Transaction" box. It would also be very helpful to have different colours for each category to spot them easier.

Apart from that, I would require a login form as right now anyone could access the application and make changes to the stored information. On a different note, the transactions we make with "suppliers" are not immediately complete but can last a longer time period. It would be nice to have a time table to see the progress.

Overall, I am very satisfied with the application so far.