***Phase 3***

***Implementation***

**Sub Contents of Implementation Page**

**11. Method of solution -**

**12. Accurate method of solution -**

**13. Programming coding -**

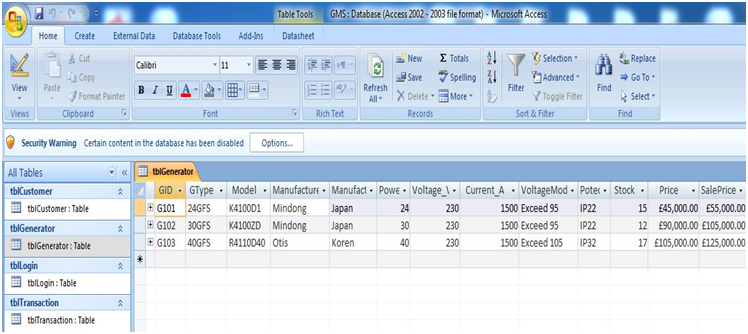
**11. Method of solution**

I used Microsoft Office Access 2007 to create the database which contained the transaction table, customer table, product table, etc.

* Click on Start. Open All Programs. Click on Microsoft Office Access2007. Create blank table

I used Microsoft Visual basic 6.0 to create the data manipulation interface.

* Click on Start. Open All Programs. Open on Microsoft Visual Basic 6.0. Click on Microsoft Visual Basic 6.0. Open **Standard EXE**.

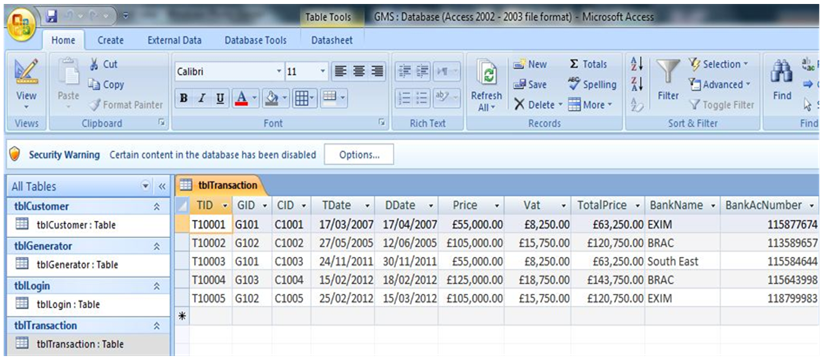
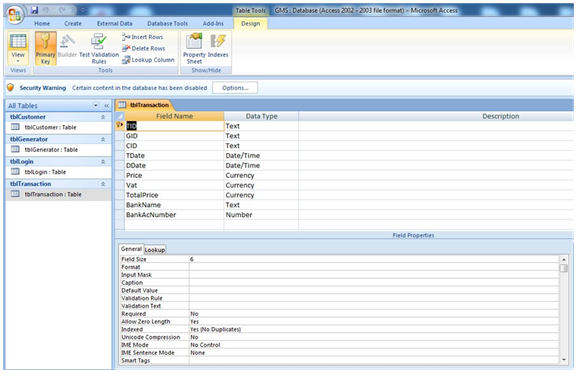


Data Type

Primary Key (No Duplication)

Generator Table Datasheet view (Fig 3.02)

Generator Table Design View (Fig 3.1**)**



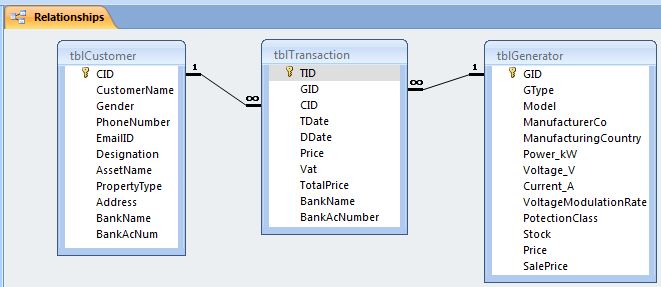
Transaction Table Datasheet View (Fig 3.4)

Data Type

Transaction Table Design View (Fig 3.3)

Primary Key (No Duplication)

Relationship Diagram (Fig 3.5)



Tool Box (Fig 3.6)



Option Button

Pointer

Text Box

Command Button

Combo Box

Adodc

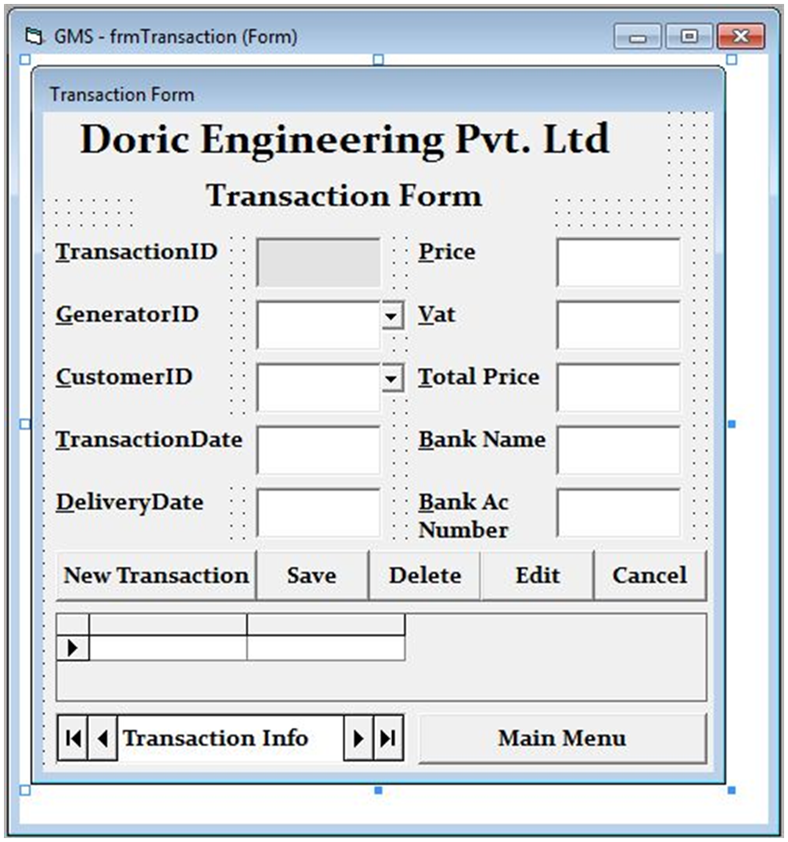
Data Grid

Data Combo

Label

Frame

Input layout of Transaction Form (Fig 3.7)



Text Box

Label

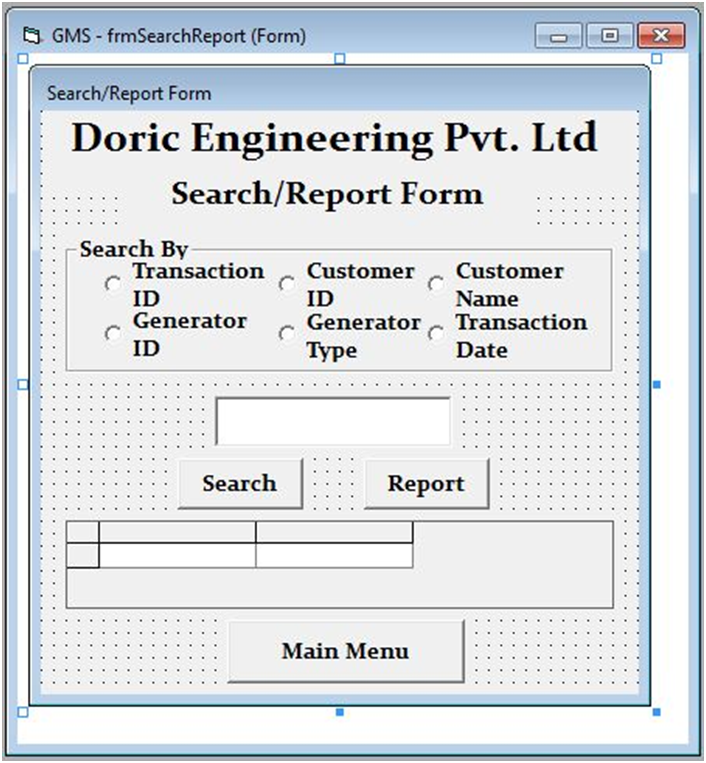
Command Button

Data Grid

Adodc

The Input layout of the **Customer Form** and the **Generator Form** is similar to the input layout of the **Transaction Form (Fig 3.7).** The **Login Form** and the **Main Form** use similar tools as the transaction from.

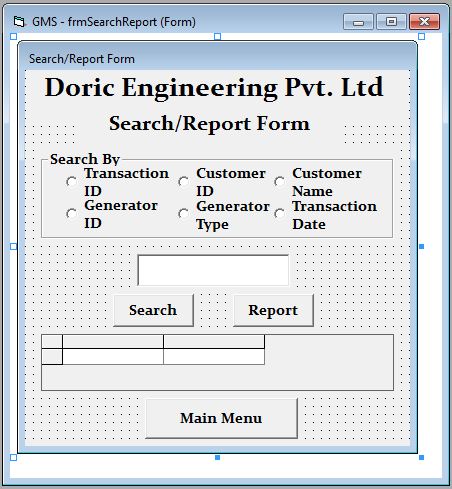
Search and Report Form (Fig 3.8)



Input Box for data query

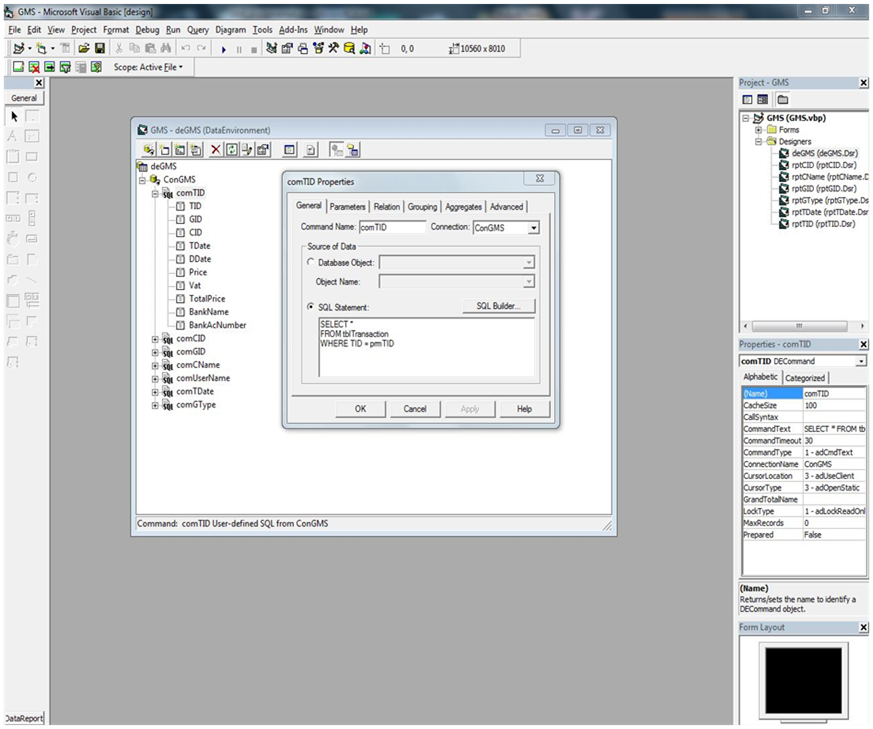
Do query by:

Frame



I used the **Search and Report Form (Fig 3.8)** to input data to refer to for searching and report. I used **Data Environment (Fig 3.9)** and the **SQL Statement (Fig 3.10)** according to the query requirement to connect to the table in the database to enable query for data. And also, the report option is available to display the query as a **report (Fig 3.11)**.

Data Environment (Fig 3.9)



SELECT \*

FROM tblTransaction

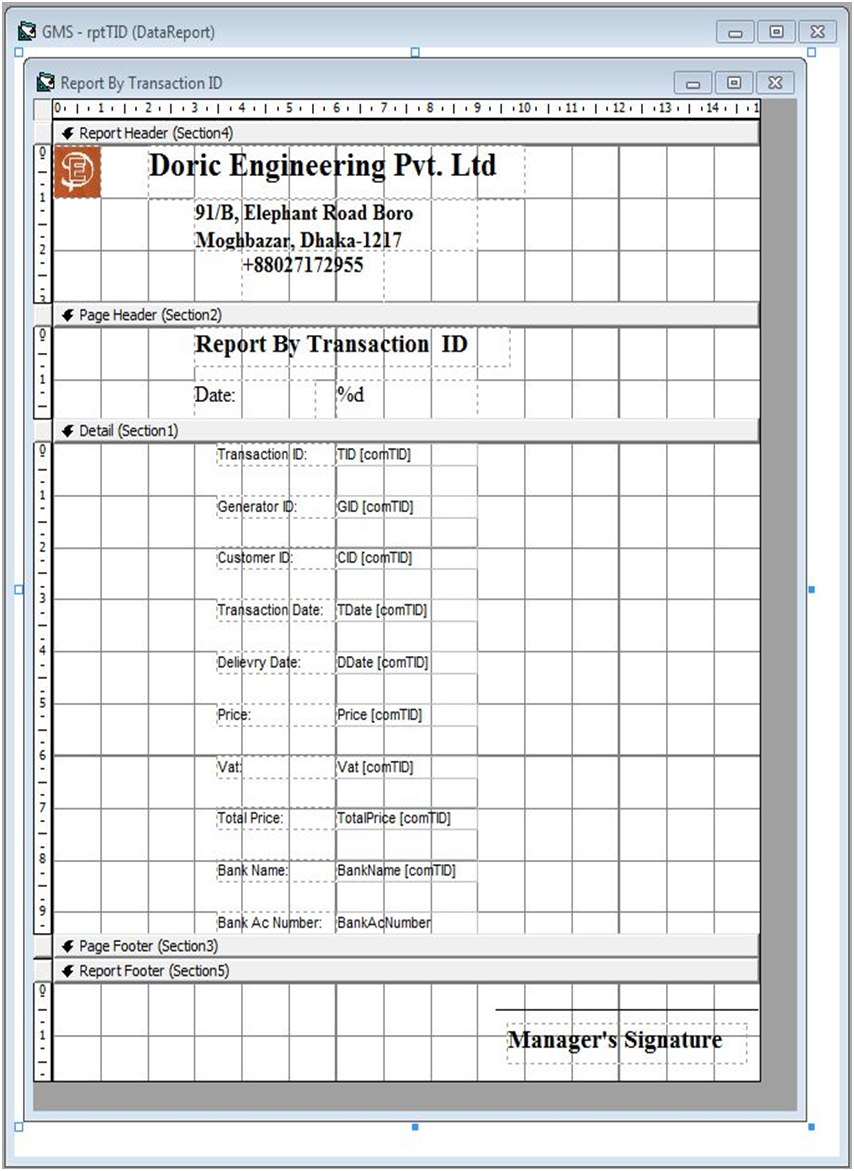
WHERE TID = prmTID

“prm TID” is the parameter for the Transaction ID based query and report.

SQL Statement Table (Fig 3.10)

|  |  |
| --- | --- |
| Command Name | SQL Statement |
| comTID | SELECT \* FROM tblTransaction WHERE TID = prmTID |
| comCID | Select \* From tblCustomer WHERE CID=prmCID |
| comGID | SELECT \* FROM tblGenerator WHERE GID=prmGID |
| comCName | SELECT \* FROM tblCustomer WHERE CustomerName=prmCustomerName |
| comUserName | SELECT \* FROM tblLogin WHERE UserName=prmUserName |
| comTDate | SELECT \* FROM tblTransaction Where TDate=prmTDate |
| comGType | SELECT \* FROM tblGenerator WHERE GType=prmGType |

Report by Transaction ID (Fig 3.11)



Data Label

Logo

Data Fields showing data extracted from the database table

The Report by **Customer ID**, **Generator ID**, etc are similar to that of the **Transaction ID (Fig 3.11).**

**12. Accurate method of solution**

According to the computer related objectives mentioned in the analysis phase and the action plan in the design phase whether those are achieved.

1. *User interface: The interface of the computerized system should be user friendly so that it is easier to use, by using navigation tools, drop-down lists, etc.*

* Has been successfully achieved by creating forms and using the tools available in the software Visual Basic 6.0. (Pg.; Fig 3.)

1. *Data storage: Huge Data should be stored in an orderly manner in separate tables for transaction, customer, etc.*

* Has been successfully achieved by creating tables in a database and storing data in them accordingly. (Pg.; Fig 3.2, 3.4)

1. *Auto ID Generation: To make the records in the table unique, like Transaction ID, Customer ID.*

- Has been successfully achieved by using the auto ID generating codes. (Pg.; Fig 3.)

1. *Avoid data duplication and redundancy: Data will not be duplicated and data redundancy must be avoided by the use of product key, relationship between primary key & foreign key, as a result consistency of data will be increased, like Customer ID used in Transaction table.*

* Has been successfully achieved by creating relationships between tables using primary keys. (Pg.; Fig 3.5)

1. *Validation: There should be validation and verification checks to reduce the errors in data.*

* Has been successfully achieved by using the validation codes (Pg.; Fig 3.)

1. *Search & Reports: The new system should provide faster data searching and faster data reports with much less human intervention.*

* Has been successfully achieved by creating a **Search and Report Form** (Pg.; Fig 3.8), **Data Environment** (Pg.; Fig 3.9) and the **SQL Statements** (Pg.; Fig 3.10)using Visual Basic 6.0.

1. *Security: There should be a security system to prevent unauthorized people from viewing confidential data.*

* Has been successfully achieved by using the **Login Form (**Pg.; **Fig 3.)** security codes in (Fig 3.)

**13. Programming Code**

Data Manipulation

Adding a new record (Fig 3.12)

|  |  |
| --- | --- |
| addnew.JPG | |
| Code | Purpose |
| adcTransaction.Recordset.MoveLast  xTID = Right(txtTID, 5)  adcTransaction.Recordset.AddNew  txtTID = "T" + CStr(xTID + 1) | For adding a new record. Then performing auto ID generation - first by taking the last records numeric code as “xTID” and then adding “1” to it and then adding it as a text next to “T”. |
| txtTDate.Text = Date | To set the current date as the “Transaction Date”. |
| txtDDate.Text = DateAdd("d", 15, Date) | To add 15 days to the Transaction Date, this will be the Delivery Date of the product. |
| MsgBox "Record Added Successfully", vbInformation, "New Transaction" | To give a message to the user that the record was added without any problems or errors. |

Editing a record (Fig 3.13)

|  |  |
| --- | --- |
| edit.JPG | |
| Code | Purpose |
| txtTID.Locked = False  txtGID.Locked = False  dcmbGID.Locked = False  txtCID.Locked = False  dcmbCID.Locked = False  txtTDate.Locked = False  txtDDate.Locked = False  txtPrice.Locked = False  txtVat.Locked = False  txtTPrice.Locked = False  txtBName.Locked = False  txtBAcNum.Locked = False | To allow all the fields to be edited, by unlocking them the field lock. |
| MsgBox "Data Edit Has Successfully Been Enabled", vbInformation, "Edit....." | To give a message to the user that the record can now be edited. |

Deleting a record (Fig 3.14)

|  |  |
| --- | --- |
| delete 1.JPG | delete 2.JPG |
| Code | Purpose |
| d = MsgBox("Are you sure that you want to delete?", vbYesNo, "Delete.....") | “d” is used as a variable to store the decision taken when the delete query was made.  A decision box is shown to ask the user to confirm his “delete action”. |
| If d = vbYes Then  adcTransaction.Recordset.Delete  adcTransaction.Refresh  Else:  Exit Sub  End If | To delete the selected record if the decision was “Yes”, else exit from the running from function. |
| MsgBox "Record Deleted Successfully", vbInformation, "Deleted....." | To give a message to the user that the record has been deleted. |

Saving a record (Fig 3.15)

|  |  |
| --- | --- |
| save.JPG | |
| Code | Purpose |
| adcTransaction.Recordset.Update | To update the all the fields/ amended fields in the Transaction Table. |
| s=adcGenerator.Recordset.Fields("Stock").Value  s = s - 1  With adcGenerator.Recordset  .Fields("Stock") = s  .Update  End With | To subtract “1” from the stock of the selected generator.  To update the “Stock” field in the Generator Table. |
| MsgBox "Record Saved Successfully", vbInformation, "Saved....." | To give a message to the user that the record has been saved. |

Data Searching and Report

Searching for a record (Fig 3.16)

|  |  |
| --- | --- |
| search.JPG | |
| Code | Purpose |
| If optTID.Value = True Then  txtSearchReport.SetFocus | To do the following query functions if it is selected and focus the cursor in the SearchReport text box. |
| If deGMS.rscomTID.State = adStateOpen Then  deGMS.rscomTID.Close  End If | To close the data environment (deGMS) object if it is open, to avoid unexpected errors. |
| deGMS.comTID Trim(txtSearchReport.Text) | To search using the SQL code in the selected command in the data environment (deGMS ) using the query request in the SearchReport text box. |
| grdSearch.DataMember = "comTID" | To display the search result in the Search Grid (grdSearch). |
| MsgBox "Search Result Shown", vbInformation, "Search" | To give a message to the user that the query for the record has been successful. |

Data Report (Fig 3.17)

|  |  |
| --- | --- |
| Transaction Report run.JPG | |
| Code | Purpose |
| If optTID.Value = True Then  txtSearchReport.SetFocus | To do the following report function and focus the cursor in the SearchReport text box. |
| If deGMS.rscomTID.State = adStateOpen Then  deGMS.rscomTID.Close  End If | To close the data environment object if it is open, to avoid unexpected errors. |
| deGMS.comTID Trim(txtSearchReport.Text) | To make a report using the SQL code in the selected command in the data environment (deGMS ) using the query request in the SearchReport text box. |
| rptTID.Show | Show the Transaction Report, such as the above. |

Validation

Length Check (Fig 3.18)

|  |  |
| --- | --- |
| length check.JPG | |
| Code | Purpose |
| If Len(txtCName.Text) > 25 Then  MsgBox "Customer Name cannot be more than 25 characters", vbCritical, "Error"  End If | Used in the Customer Name field in the Transaction Form to check the name input if the length is more than 25 characters and shows error message in case the data is more than 25 characters. |

Format Check (Fig 3.19)

|  |  |
| --- | --- |
| format check.JPG | |
| Code | Purpose |
| If KeyAscii = 13 Then  If txtTID Like "T#####" Then  Else  MsgBox "Wrong Format T#####", vbCritical, "Error Input"  End If  End If | Used to check if the Transaction ID is in the correct format or not and shows error message in case the data is in the wrong format. |

Type Check (Fig 3.20)

|  |  |
| --- | --- |
| type check.JPG | |
| Code | Purpose |
| If IsNumeric(txtPhNum.Text) = True Then  Else: MsgBox "Phone Number can be numeric only", vbCritical, "Error"  txtPhNum.Text = ""  txtPhNum.SetFocus  End If | Used in Phone Number text box in the Customer Form to check if data is only numeric and shows error message if data of any other type is present. |

Presence Check (Fig 3.21)

|  |  |
| --- | --- |
| presense chk2.JPG | presense chk.JPG |
| Code | Purpose |
| If KeyAscii = 13 And txtGID = "" Then  MsgBox "Empty Field", vbCritical  txtGID.SetFocus  ElseIf KeyAscii = 13 Then  txtCID.SetFocus  End If | Used to check the field "Generator ID" in the Transaction Table to find if the field contains null value and prevents from moving the cursor to the Customer ID field if data is not present and also shows error message in case the data is not present. |

To see other workings of Data Manipulation, Search and Report, Validation including other Forms see Testing Phase. (Page -)