Contents

Chapter 1

Analysis

1.1 Introduction

1.1.1 Background

I am creating this system for Sunny Future Solar, a trading name for Sunny Future Ltd., a Company based in Aldershot, Hampshire, United Kingdom. As its name suggests, the Company installs solar panels, more specifically solar photovoltaic panels, but it is a very small Company, having only two full time employees who are also its directors: Philip and Angi Long. The Company is accredited by the regulation body for all renewable technologies, the Microgeneration Certification Scheme (MCS).

1.1.2 Problem definition

Sunny Future Solar must comply with the stringent documentation regulations laid out by the MCS. This involves producing many documents (invoices, quotes, delivery logs, incident logs, survey forms, etc.) and keeping them all up to date. Always having the correct versions of any one document, or indeed the same numbering system depending on who uses the computer, can be problematic as they do not follow the same personal guidelines, and one user is potentially more proficient (or patient!) than the other. The current system does not have the ability to search documents or store central copies and have customer details stored. The current system requires duplication of quotation and invoice data—Philip handwrites them and Angi types them up, and delivery/error log forms are filled in by hand.

1.1.3 The users

Sunny Future Ltd.'s directors and employees will make use of the system. One director, Philip Long, is the main worker in the business. He has limited IT knowledge. The second director, Angi Long, the secretary, has good knowledge of Microsoft Office, and adapts well.

1.2 Investigation of user needs and acceptable limitations

1.2.1 The current system analysis

1.2.1.1 Interview (1), 13/10/2011, 14:00

During the first interview, I asked Angi Long, one of Sunny Future Solar's directors, the following questions and received the following responses:

Q: What is the current system? What does it do? What data is input and output? How do you update it?
A: The current system has been developed from the requirements of the MCS and the REAL Consumer Code and practical situations during the last 17 months of trading. The system deals predominantly with the contact with the customer from enquiry to final invoice, but also purchase ordering and an element of stock control.

Data input into and output from the current system:

INPUT
Customer details:
Name
Address
Postcode
Telephone Number
Email Address
Installation Address (if different)
MPAN No.

INPUT (again)
Designed PV System Components:
kWp
Modules
Mounting
Inverter
Quotation No.
Net price
VAT
Total price
Required deposit on acceptance
Sap Calculation
FiT Calculations
Expected total benefit per year
Agreed Installation Date
Supplier Details
Purchase Order No.
Required Delivery Date
Goods-In
Component Serial Numbers
Invoice No.
Net price calculations

OUTPUTS
Initial Enquiry/Survey Form (Internal)
Covering Letter
Quotation
Quotation Log (Internal)
Benefit Sheet
Acknowledgement of Order
Receipt for deposit
Confirmation of Installation Date
Purchase Order
Purchase Log (Internal)
Goods-In sheet (Internal)
Final Invoice
Invoice Log (Internal)
Guarantee

Q: What are the problems with the current system?
A: Each element is currently added separately at each stage which takes time and is frustrating.

Q: What is your current IT infrastructure? Would you be prepared to purchase additional software?
A: Sunny Future Solar work on Microsoft Windows on three computers, and use Microsoft Office 2007. They would rather not spend money on software. They also use Dropbox for shared filing, as their computers are not networked.

Q: Do you have a particular solution in mind? Can you think of any possible constraints?
A: Fill in one database entry and for all the documentation, the relevant fields are completed by the database so that no repetition occurs with data entry. However, the information must be able to be edited when necessary.

1.2.1.2 Observation

- Sunny Future Solar received an enquiry.
- Angi arranged a time for Philip to visit the enquirer's house to conduct a survey of his roof and establish requirements.
- Angi filled in the customer's name, address, telephone number and email address on a printed paper copy of the survey form.
- Philip went to the house a few days later to conduct a survey. After this, I went back to the office to further observe the process.
- The customer was keen, on the strength of Philip's comments when he did the survey, so Angi entered his details into a quotation document in Word, inputting them manually from the handwritten, printed survey form.
- Philip then sat down and handwrote the rest of the quotation, doing the necessary calculations (SAP, proposed price, deposit amount etc.) and specifying components according to the customer's wishes. Angi then typed this quotation into the Microsoft Word document she had created beforehand.
- The quotation was then saved to the harddrive, implicitly backed up to Dropbox, and printed as hard copy twice: once for the customer, and once for Sunny Future Solar's customer paper folder, along with the hard copy of the survey form.
- The customer's copy of the quotation was sent to him that day.
- The customer accepted his quotation and paid his deposit a few days later, so I returned again. Then, Angi was liasing with suppliers, organising delivery of components, and had fixed an installation date with the customer.
- The customer had his system installed two weeks later, and after that Angi typed an invoice, specifying the components actually used, the installation address (which in this case was the same as the billing address), and the final total price. This was again saved and backed up, and printed twice, and sent to the customer.

1.2.1.3 Investigation of documentation

Sunny Future Solar provided me with various pieces of current system documentation. Here are my findings:

Their current documentation is paper-based, so hard to manage as multiple people, for example the two directors, cannot have multiple copies of the same piece of paper with the same data on it, so inconsistencies between numbering of entries in forms arise, for example. With regard to the MCS accreditation, Sunny Future Solar are required to keep copies of various reports, forms and strategies to comply with regulations, along with a document master list with all document numbers which is very intricate. These documents require lots of data to be input into several documents, so it is not practical to do it by hand—they need a system which will take data from one form and put it where it is meant to go in another according to certain conditions.

1.2.1.4 Investigation of input forms, output forms and report formats from the existing system

Sunny Future Solar provided me with various pieces of documentation: a quotation, an invoice, a quality plan, a potential benefits sheet, a cancellation form, a survey form, a job sheet, a purchase form, a purchased order log sheet, and the MCS master documentation list.

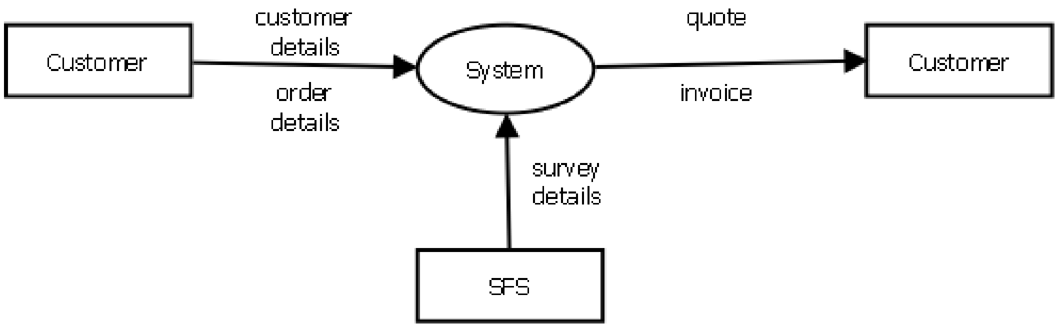
The potential benefits sheet is a sheet mostly made up of static text, just with the SAP calculation (which is different for every person, so at the moment requires manual entry) inserted.
The job sheet duplicates some of the information contained on the survey form.
The purchase form indicates the order number, the order date, the components sought, the shipping cost, and the total cost, and the delivery date. This is filled in by Sunny Future Solar after every order for components is placed.
The purchase order log sheet is used to record when Sunny Future Solar receives the shipment.

1.2.2 Data Flow Diagram of the current system

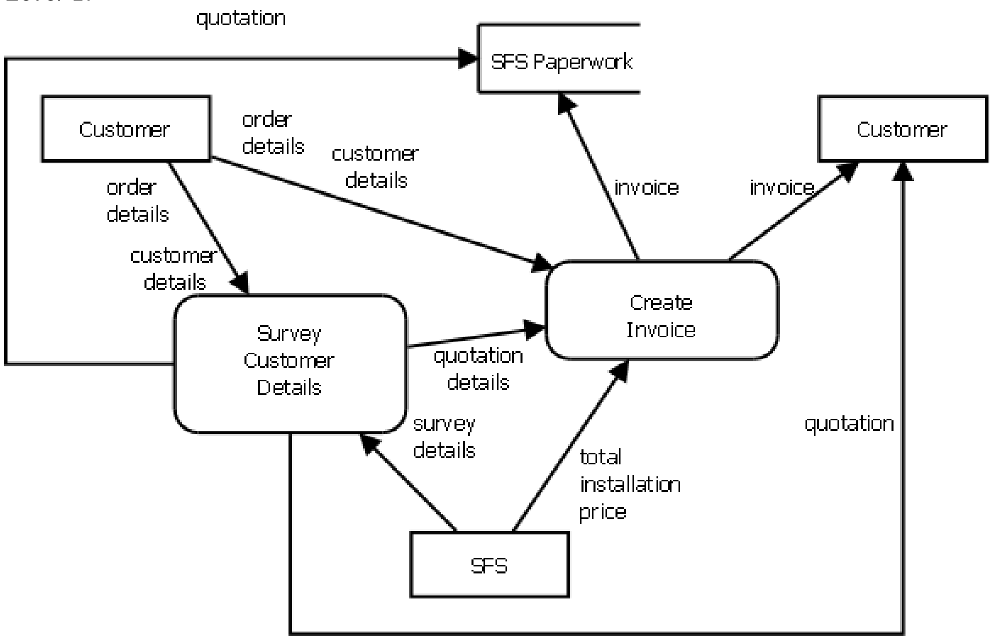
Here are the data flow diagrams (level 0 to level 1) that I devised based on the process documents for an installation plan that Sunny Future Solar provided.

In these diagrams and 'Data sources and destinations', Sunny Future Solar is referred to as 'SFS'.

Level 0:



Level 1:

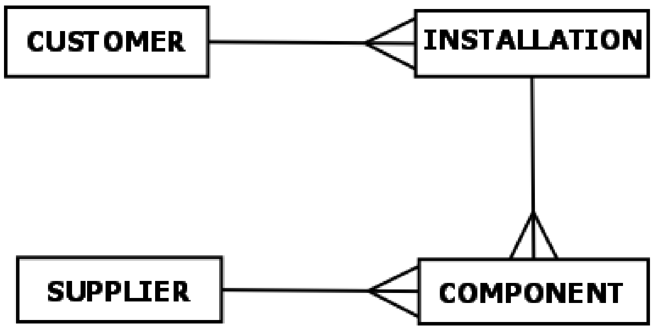


1.2.3 Data sources and destinations

The first data source is the customer: he or she gives the order details and his or her personal details (address, name, email address. . .) to SFS, which goes onto a survey form. The survey form details are used by SFS to produce a quotation which gets printed and stored in the SFS paperwork customer folder, and also posted to the customer. The quotation, as well as being an output, is also used as a data source for the invoice after the job is done: component details and customer details are taken from that quotation, adapted if necessary depending on what was really installed, and again stored in SFS's paper customer folder, and printed and sent to the customer for payment and their own records. SFS is also a data source as they provide component details frmo their suppliers, and price and deposit details.

1.2.4 Entity relationship diagram and entity descriptions of the current system

Paper/brain-based.



Every customer can have many installations; Each installation can only be installed on one customer's roof. Every installation can have many components. Each component can only be fitted onto one installation. Every supplier can supply many components. Each component can only be supplied by one, specialist supplier (the best one).

1.2.5 Discussion of problems with the current system

The current system is fragmented, open to data entry duplication and error, and not sufficiently efficient for a successful solar panel installation business.

1.2.6 The proposed new system analysis

1.2.6.1 User needs

Much of the data included on one paper form (a quotation, for instance, or an order job sheet) is repeated—name, address, calculations and panel details are examples. This data needs to be stored in a centralised database so that the data is not repeated unnecessarily due to the fact that Sunny Future Solar operates from three different computers: one desktop in their office, plus two laptops. The user needs not to waste too much time so that he or she can spend more time finding customers, and the fewer minutes or hours he or she wastes entering data, the fewer times he or she has to enter it, logically, so lessening the number of potential data entry errors. Reports have to be created: quotations, invoices, logs and survey forms. These will require Microsoft Word compatibility and printing.

1.2.6.2 Interview (2), 24/10/11, 19:00

Q: What should the new system actually do?
A: The new system requires the creation of a database to record the information above. It should also hold write only 'field' versions of the paperwork required, automatically completed by the database record ready for use.

Q: What reports are required—would you like it to work with Word?
A: Reporting is required: logs are completed (see above), along with invoices and quotations. It would need to work with Microsoft Word (oepning, editing, etc.).

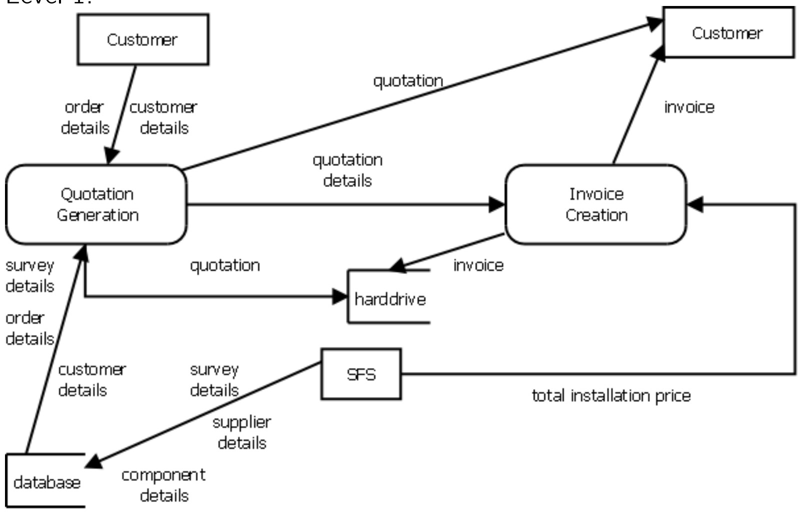
Q: How much data will the system hold per section?
A: A potential maximum of 500 records per year.

1.2.7 Data Flow Diagram of the proposed new system

Here is the level 1 data flow diagram that I devised based on my research and investigation into what Sunny Future Solar's new system will be.

In this diagram, Sunny Future Solar is referred to as 'SFS'.

Level 1:

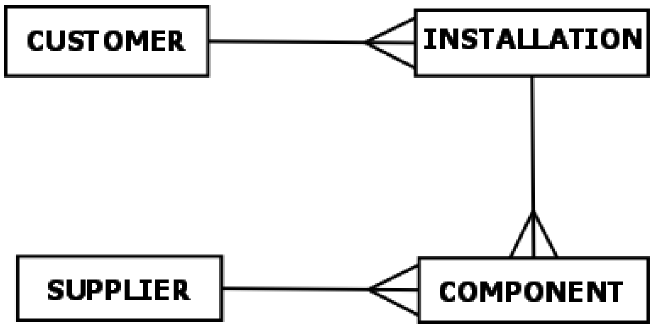


1.2.8 Data sources and destinations

The first data source is the customer: he or she gives the order details and his or her personal details (address, name, email address. . .) to SFS, which goes onto a survey form. The survey form details are used by SFS to produce a quotation which gets printed and stored on the computer's harddrive, and also posted to the customer. The quotation, as well as being an output, is also used as a data source for the invoice after the job is done: component details and customer details are taken from that quotation, or the database, adapted if necessary depending on what was really installed, and again stored in SFS's computer file system, and printed and sent to the customer for payment and their own records. SFS is also a data source as they provide component details frmo their suppliers, and price and deposit details, though these are in the database via the program.

1.2.9 Entity relationship diagram and entity descriptions of the new system

Computer/database-based.



Every customer can have many installations; Each installation can only be installed on one customer's roof.
Every installation can have many components. Each component can only be fitted onto one installation.
Every supplier can supply many components. Each component can only be supplied by one, specialist supplier (the best one).

Customer(cust_id, cust_title, cust_name, cust_billaddress, cust_billpostcode, cust_instaddress, cust_instpostcode, cust_hometelno, cust_mobtelno, cust_mpan, cust_email)
Supplier(supp_id, supp_name, supp_address, supp_postcode, supp_telno, supp_contactname)
Component(comp_id, comp_name, comp_type, comp_serialno, comp_panelwp, comp_supplier*)
Installation(inst_id, inst_netprice, inst_vat, inst_totalprice, inst_deposit, inst_dateinstall, inst_sapcalc, inst_purchordernum, inst_deliverydate, inst_invoiceno, cust_id*)

This database is in the first normal form (1NF) because every piece of data is atomic, i.e. there are no repeating groups.
The database is in the second normal form (2NF) because it is in the first normal form (1NF) and there are no partial key dependencies.
The database is in the third normal form (3NF) because it is in the second normal form (2NF) and contains no non-key dependencies.

1.2.10 Analysis data dictionary

Customer:

Item Name	Data Type	Size (chars)	Example
Surname	String	25	Long
Forename	String	22	Isabell
Address	String	100	12 Reputable Place
TelNum	String	13	01234 567890
Email	String	25	example@example.com
InstallAddress	String	100	54 SolarInstall Place
MPANNumber[1]	Integer	32 (bits)	00012345678901

[1]The MPAN number is the number that uniquely identifies every electricity supply point (individual domestic, for example) in the UK.

Component:

Item Name	Data Type	Size (chars.)	Example
PanelName (module)	String	20	Sanyo
PanelType (module)	Integer	32 (bits)	235
Watts	Integer	32 (bits)	3250
Mounting (frame)	String	20	Renusol
InverterName	String	15	SMA
InverterType	String	20	SB1700TL
SerialNumber	String	10	SEN1024854
BracketType	String	10	Bracket001

Supplier:

Item Name	Data Type	Size (chars.)	Example
SupplierName	String	25	Alternergy
SupplierAddress	String	100	25 Supplier Place
SupplierTelNo	String	13	01234 567890
SupplierContactName	String	25	Alex Wright

Installation:

Item Name	Data Type	Size (chars.)	Example
QuotationNumber	Integer	32 (bits)	001234
NetPrice (£)	Integer	32 (bits)	10000
VAT (%)	Integer	32 (bits)	5
TotalPrice (£)	Integer	32 (bits)	15000
DepositAmount (£)	Integer	32 (bits)	500
AgreedInstallDate	Date	10	03/04/2012
SAPCalc[2]	String	20	$0.8 \times 2.9 \times 1073 \times 0.8$
PurchaseOrderNum	Integer	32 (bits)	12345678909876543210
DeliveryDate	Date	10	02/03/2012
InvoiceNumber	Integer	32 (bits)	001234

1.2.11 Data volumes

Sunny Future Solar estimate a maximum of five hundred customers per year, so a maximum of five hundred customers will be inserted into the Customer table. Sunny Future Solar alternate between suppliers, but have seven main suppliers at present, therefore a theoretical maximum of thirty suppliers in the supplier table. As for

[2]The SAP calculation is the calculation devised by the government to ensure that solar installation companies do not over extol the Feed in Tariff benefits of solar PV.

components, many, many components exist, not all of them available at the same time; therefore, a realistic estimate for components (from the thirty suppliers) is six hundred, for the component table (600/30).

1.3 Constraints

1.3.1 Hardware constraints

Sunny Future Solar operate from three reasonably new computers: two laptops and a desktop in their office. Therefore, the computer that this program will be running on will be up-to-date enough to handle it.

1.3.2 Software constraints

All Sunny Future Solar's computers are equipped with versions of Microsoft Windows, either Vista or 7, and Microsoft Office 2007, so there will be no problem with software versions.

1.3.3 Time constraints

The system has to be built in months, not years: precisely, I have until April 2012 to build the system. Due to this, I may not have time to implement all the features that my user would like: this analysis has been an attempt at creating realistic targets.

1.3.4 User's knowledge of information technology

The users have reasonable knowledge of information technology: they use Microsoft Windows computers on a daily basis, and are proficient with Microsoft Word.

1.3.5 Who will be allowed to use various parts of the system

Philip and Angi Long will be allowed to use the system, and future employees will be able to access certain parts of the system if authorised by Philip or Angi to do so.

1.4 Limitations

1.4.1 Areas which will not be included in computerisation

- Implementation of a stock control system.

1.4.2 Areas considered for later development

- Implementation of a stock control system.
- Cross-platform compatibility.

1.5 Objectives

1.5.1 General objectives

- Streamline the efficiency of the administration process within Sunny Future Solar.

1.5.2 Specific objectives

1. Have a main menu that allows the user to select different options.
2. Have a functioning relational database, queriable with runtime SQL, made in Microsoft Access, with the required number of tables.
3. Enable the user to add customers.
4. Enable the user to add suppliers.
5. Enable the user to add components.
6. Enable the user to remove customers.
7. Enable the user to remove suppliers.
8. Enable the user to remove components.
9. Enable the user to list customers.
10. Enable the user to list suppliers.
11. Enable the user to list components.
12. Enable the user to view and edit invoices, log forms, and reports in Microsoft Word by clicking buttons in the program to open the requested forms.
13. Enable the user to view relationships between suppliers and the components they stock.
14. The program must input data into the invoices and quotes to avoid the user having to duplicate data entry, and this data must come from either user input or data from the database.
15. The system should be menu-driven, with consistent GUI form layout throughout, as far as possible.
16. Enable printing directly from the program for the user to print lists of customers etc.
17. Enable searching of customers, suppliers and components, and sort the search results.

1.6 Consideration of alternative solutions

The proposed solution is to write the system in Visual Basic.NET with a Visual Basic forms GUI frontend, a Microsoft Access database, and Microsoft Office integration for the reports. An alternative solution would be to not build a bespoke system but use an already built system such as SAGE, but, for Sunny Future Solar who have minimal amounts of data to input and manipulate, such a big and complex system would take up unnecessary space and cost unnecessarily large amounts of money. Another alternative solution would be to code the solution in a language other than VisualBasic.NET, enabling cross-platform compatibility, however Visual Basic is the language I am most familiar with, and no operating systems other than Windows run on any of Sunny Future Solar's office computers.

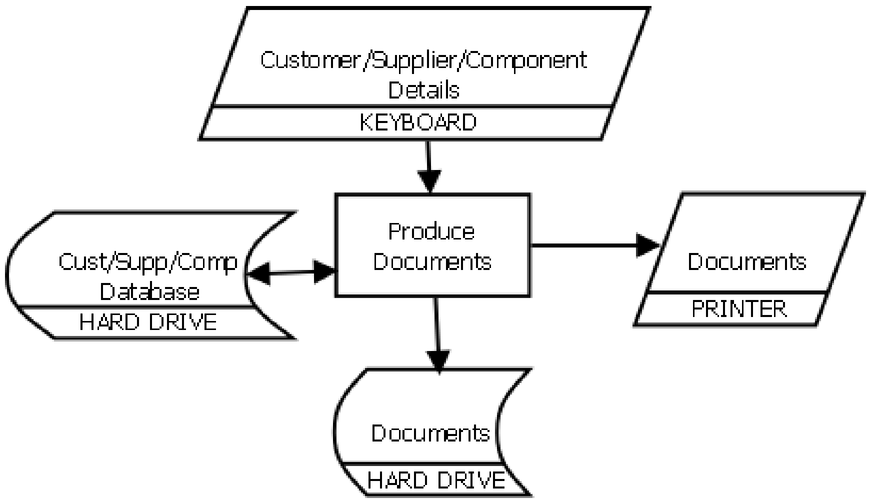
1.7 Justification of the chosen solution

Building the system in VisualBasic.NET is best because a fully documented system will be available and the developer of the system will be available to fix the system if it goes wrong, which would not be possible if they were to use an already built package developed by someone remote as mentioned above. Also, features will be able to more easily be added as and when the user wants them, rather than having to go through a potentially slow process of emails and phone calls to invisible people. VisualBasic.NET is the language I feel most comfortable with, and Sunny Future Solar require easy integration with Microsoft Word which works well with VB.NET.

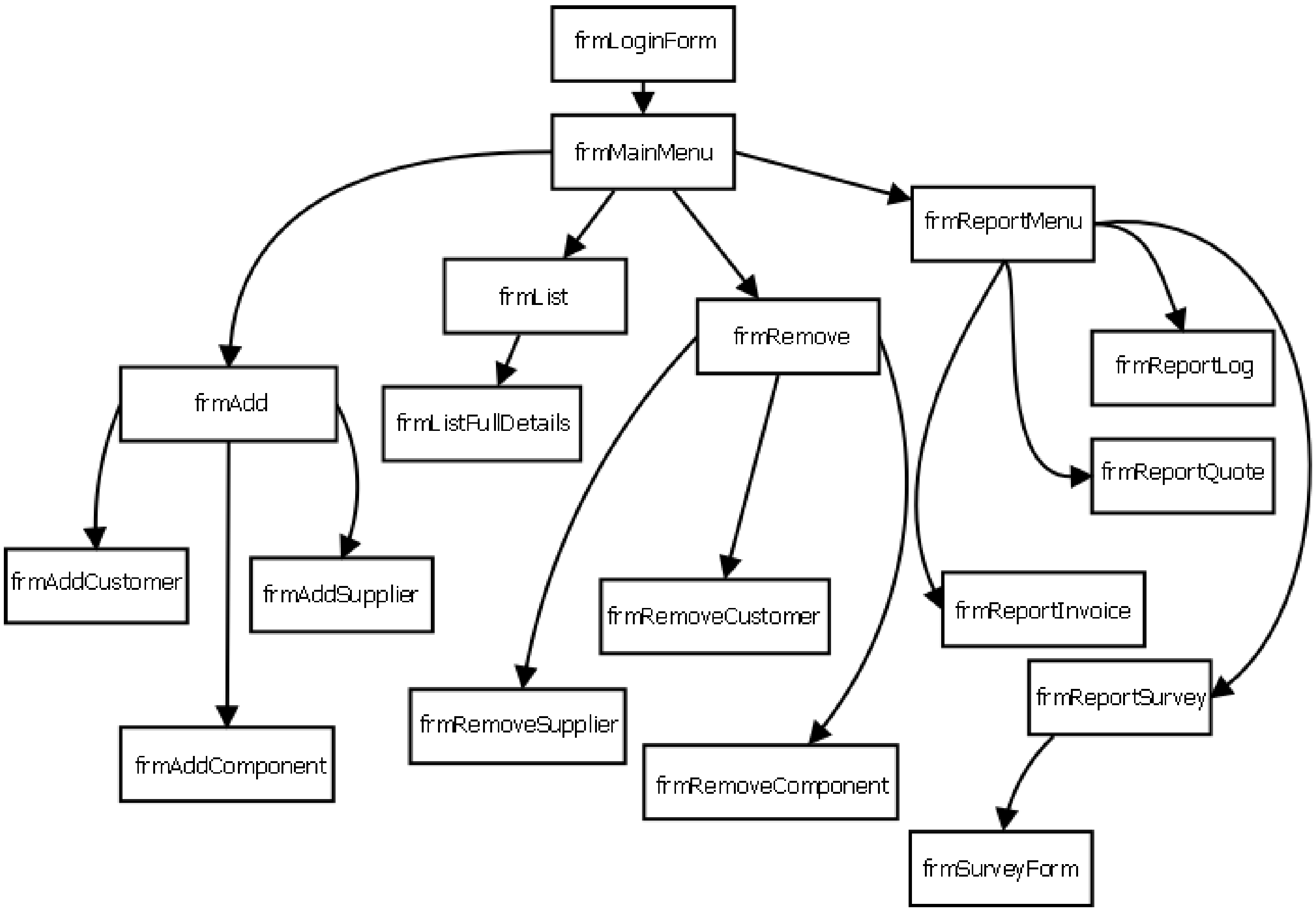
Chapter 2

Design

2.1 Overall system design



2.2 Description of modular system structure



Input	**Process**	**Storage**	**Output**
Customer data	Validation	Customer table in the database	
Supplier data	Validation	Supplier table in the database	
Component data	Validation	Component table in the database	
Survey details	Validation	Installation table in the database	
Selected customer	Microsoft Word opens, the program having executed code to make the quotation or invoice	A Microsoft Word document	A Microsoft Word document containing customer details
Selected log form	Microsoft Word opens, the program having executed code to open the selected log form	A Microsoft Word document	A Microsoft Word document containing the various log tables

2.3 Design data dictionary

Customer:

Field Name	**Example**	**Data Type**	**Size (chars.)**	**Validation**	**Default Value**	**Key Field**
cust_id	1	Autonumber	4	Filled automatically	Increment by 1 as customers are added	Primary key
cust_title	Mr	String	4	$\neq$ NULL		
cust_name	Joe Bloggs	String	50	$\neq$ NULL		
cust_billaddress	25 BillingAddress Place	String	50	$\neq$ NULL		
cust_billpostcode	JE49 8RH	String	8	$\neq$ NULL		
cust_instaddress	26 InstallAddress Place	String	50	$\neq$ NULL if $\neq$ BillAddress		
cust_instpostcode	IP38 9FJ	String	8	$\neq$ NULL if $\neq$ BillPostcode		
cust_hometelno	01234 567890	String	12	$\neq$ NULL		
cust_mobtelno	07890 098765	String	12	$\neq$ NULL		
cust_email	x@y.com	String	25	Must contain '@'		
cust_mpan	12345789012334	String	14	Erroneous if length $\neq$ 14		

Supplier:

Field Name	**Example**	**Data Type**	**Size (chars.)**	**Validation**	**Default Value**	**Key Field**
supp_id	1	Autonumber	4	Filled automatically by the database	Increment by 1 as suppliers are added	Primary key
supp_name	Alternergy	String	25	≠ NULL		
supp_address	25 Supplier Place	String	50	≠ NULL		
supp_postcode	N1 8YE	String	15	≠ NULL		
sup_telno	03739 974636	String	13	≠ NULL		
supp_contactname	Joe Bloggs	String	25	Can = 0		

Component:

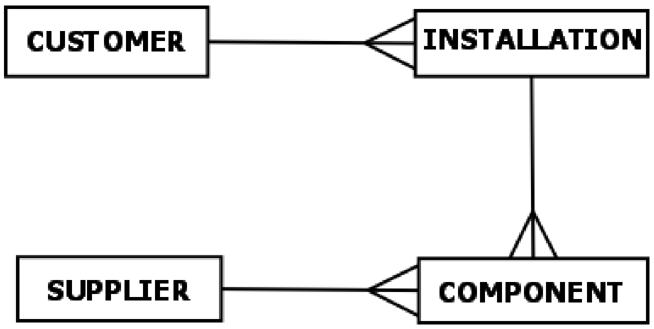
Field Name	**Example**	**Data Type**	**Size (chars.)**	**Validation**	**Default Value**	**Key Field**
comp_id	1	Autonumber	4	Filled automatically in the database	Increment by 1 as components are added	Primary key
comp_name	Suntech 250	String	50	≠ NULL		
comp_type	Solar Panel	String	50	≠ NULL		
comp_serialno	SEN1023854	String	10	≠ NULL		
comp_panelwp	3.25	Integer	32 (bits)	≠ NULL, type check: integer		
comp_supplier					Foreign key, filled from supplier table	Foreign key

Installation:

Field Name	**Example**	**Data Type**	**Size**	**Validation**	**Default Value**	**Key Field**
inst_id	1	Autonumber	4	Filled automatically	Increment by 1 as installation details are added	Primary Key
inst_quoteno	5	Integer	32 (bits)	Filled automatically	Increment by 1 every time a quote is created	
inst_netprice	8 000	Integer	32 (bits)	≠ NULL, type check: integer		
inst_vat	5.00%	Integer	32 (bits)		5.00%, type check: integer	

Field Name (cont.)	Example (cont.)	Data Type (cont.)	Size (cont.)	Validation (cont.)	Default Value (cont.)	Key Field (cont.)
inst_totalprice	15 000	Integer	32 (bits)	≠ NULL, type check: integer		
inst_deposit	1 000	Integer	32 (bits)	≠ NULL, type check: integer		
inst_dateinstall	04/01/12	Date	10	Can be NULL if customer doesn't agree to install		
inst_sapcalc	0.8×2.9×1073×0.8	Integer	32 (bits)	≠ NULL		
inst_purchordernum	1230201	Integer	32 (bits)	≠ NULL, type check: integer		
inst_deliverydate	03/01/12	Date	10	≠ NULL		
inst_invoiceno	1	Integer	32 (bits)	Filled automatically	Increment by 1 every time an invoice is created	
cust_id					Foreign key, filled from customer table	Foreign key

2.4 Database design



Every customer can have many installations; Each installation can only be installed on one customer's roof.
Every installation can have many components. Each component can only be fitted onto one installation.
Every supplier can supply many components. Each component can only be supplied by one, specialist supplier (the best one).

Customer(cust_id, cust_title, cust_name, cust_billaddress, cust_billpostcode, cust_instaddress, cust_instpostcode, cust_hometelno, cust_mobtelno, cust_mpan, cust_email)
Supplier(supp_id, supp_name, supp_address, supp_postcode, supp_telno, supp_contactname)
Component(comp_id, comp_name, comp_type, comp_serialno, comp_panelwp, comp_supplier*)
Installation(inst_id, inst_netprice, inst_vat, inst_totalprice, inst_deposit, inst_dateinstall, inst_sapcalc, inst_purchordernum, inst_deliverydate, inst_invoiceno, cust_id*)

This database is in the first normal form (1NF) because every piece of data is atomic, i.e. there are no repeating groups.
The database is in the second normal form (2NF) because it is in the first normal form (1NF) and there are no partial key dependencies.
The database is in the third normal form (3NF) because it is in the second normal form (2NF) and contains no non-key dependencies.

2.5 File organisation and processing

The system requires a Microsoft Access database to work, and various reports to produce the invoices, quotes and log sheets in Microsoft Word document files. These reports, however, cannot be deleted unless they can be easily reproduced with the same data, but the size of each report is going to be more or less the same—just documents with content but no headed paper as it will get printed onto headed paper.

The maximum file storage requirement per year for the database would be 10 000 KB. (Formula: The average row will take up 20KB of space, and SFS estimate 500 records per year, so a maximum of 10 000 KB per year.)
The maximum file storage requirement per year for the reports would be 25 000 KB. (Formula: The average quotation is 25 KB in size, with both a quotation and an invoice for each of the 500 customers, so a maximum of 25 000 KB per year.)

2.6 Identification of storage media

Stored on one computer's hard drive, and the database files will be backed up regularly to an external hard drive as well as daily and automatically to Dropbox, when changed. I am choosing a hard drive as a storage medium as the files are quite small and the desktop hard drive that SFS have is big enough. The database only has to be stored on one computer as the program is only going to be used from one computer, and the documents produced can be easily emailed or put into Dropbox sharing if they need to be shared between Philip or Angi or any other eventual employees.

2.7 Identification of suitable algorithms for data transformation, and pseudocode of those algorithms

Validation:

```
  **if** $NameTextBox \leftarrow$ "" **then** $MsgBox("Enter\ a\ name!")$
  **end if**
```

Opening a Microsoft Word document:

```
  **if** $dateselection \leftarrow$ "$December$ 2011" **then**
```

```
    $docopened \leftarrow Open("log\_dec11.docx")$
  end if
```

Bubble sort for the customers, also implementable for suppliers and components by changing 'cust_name' to 'supp_name' and 'comp_name' respectively:

```
  repeat
    $swapped \leftarrow false$
    for $i \leftarrow 0 \rightarrow length(cust\_name) - 1$ do
      if $cust\_name[i-1] > cust\_name[i]$ then
        $swap(cust\_name[i-1], cust\_name[i])$
        $swapped \leftarrow true$
      end if
    end for
  until $swapped \leftarrow false$
```

Some SQL:

Selecting all customers:

```
        SELECT * FROM Customer
```

Search for a customer:

```
SELECT cust_name FROM Customer WHERE cust_name LIKE '%" & search_text "%''
```

Insert customer details:

```
        INSERT INTO Customer (cust_title, cust_name, cust_billaddress,
        cust_billpostcode, cust_instaddress, cust_instpostcode,
        cust_hometelno, cust_mobtelno, cust_mpan, cust_email) &
        VALUES (@cust_title,@cust_name,@cust_billaddress,
        @cust_billpostcode,@cust_instaddress,@cust_instpostcode,
        @cust_hometelno,@cust_mobtelno,@cust_mpan,@cust_email)
```

Delete a supplier:

```
DELETE * FROM Supplier WHERE supp\_name = " & selectedname & ""
```

2.8 Class definitions

Not applicable.

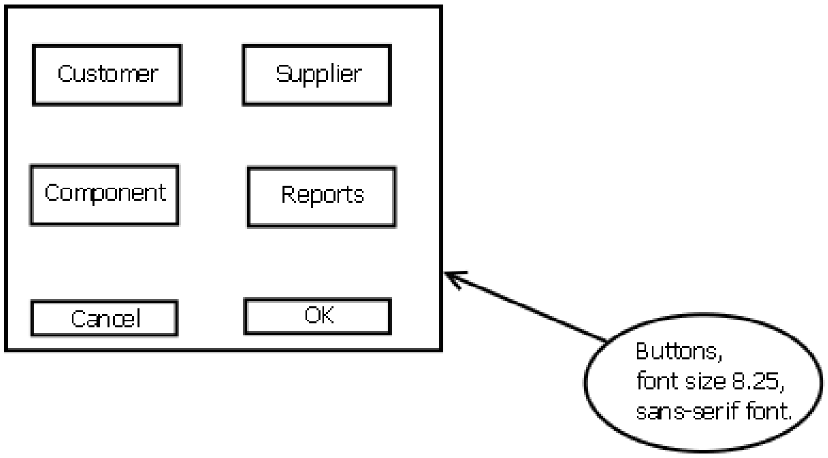
2.9 User interface rationale

- Graphical user interface:
  - Buttons.
  - Drop down menus for finite choices that the user can select, so that the user does not have to remmeber the right names for things like 'Customer' vs. 'Customers' in some places, to enable easy selection.
  - Textboxes lined up with labels, for easy data entry.

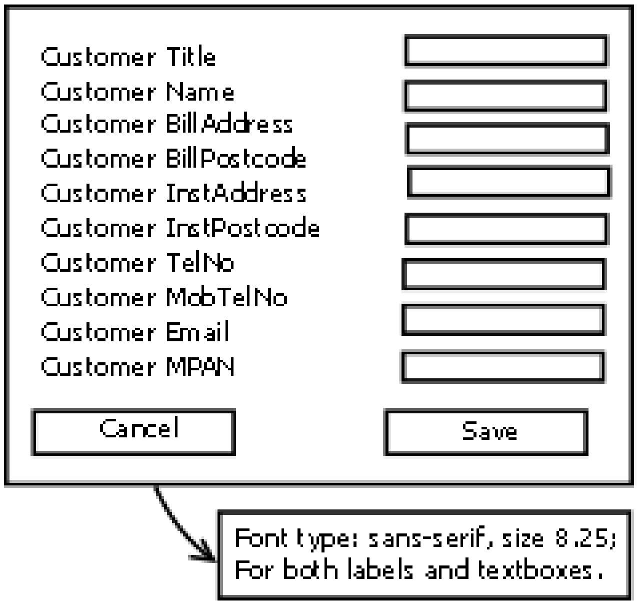
- Font:
  - Forms: sans-serif, size 8.5. Sans-serif fonts are easier on the eye and look professional.
  - Report documents: Calibri, size 22 for the headers, size 11 for the main text. Sans-serif fonts are easier on the eye and look professional.
- Colours: grey, black, blue. I am using these colours as they keep the program clear, simple and easy to read and use.

2.10 User interface sample of planned data capture and data entry designs

The main menu:

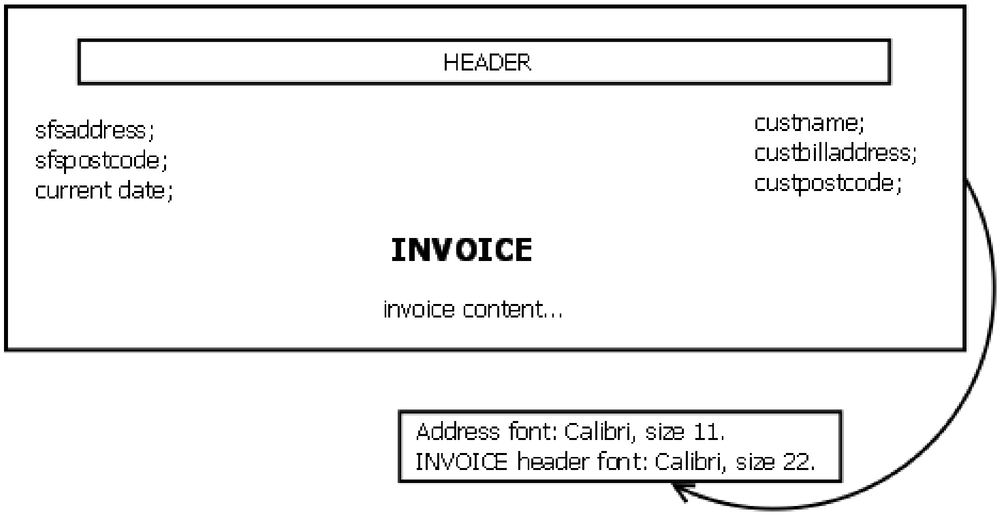


The customer entry form:

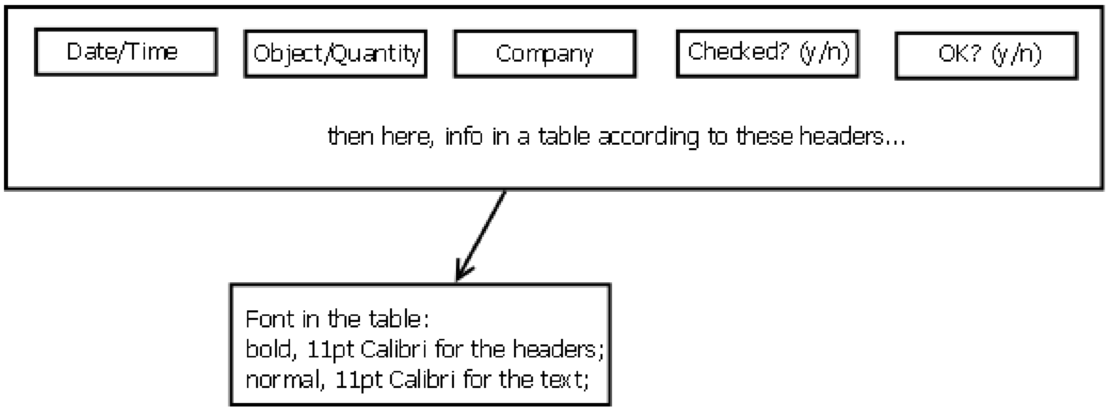


2.11 User interface sample of planned output designs

Invoice:



Log form:


Date/Time
Object/Quantity
Company
Checked? (y/n)
OK? (y/n)
then here, info in a table according to these headers...
Font in the table:
bold, 11pt Calibri for the headers;
normal, 11pt Calibri for the text;

2.12 Description of measures planned for security and integrity of data

If the user's computer crashed, they would be able to restore program files from a recent backup. These backups would be taken daily in accordance with SFS's file backup procedure, onto an external harddrive, and to Dropbox. Every user will have access to all parts of the system as all parts are important to each of the (two) users. The program itself will be password protected with a login screen, and the database containing the login details stored in a location not immediately obvious—the user will not need to look at the database unless he or she forgets his or her password. The files—invoices, quotations and log forms can be edited by any user of the computer who has write access to the directory they are stored in—usually only Angi would edit them though.

2.13 Descriptions of measures planned for systems security

One computer will contain the database and only one person will be using the system at any one time, because only one person will be in front of the computer that it will be installed on typing. If the system was to be used on multiple computers, multiple copies of the database would exist and no-one would know which was the current version, so they would get very confused. In terms of systems security, the computer and the program itself will be password protected. If the computer dies completely or burns down, Dropbox will receive copies of the data files so they can be restored on the new computer that will have to be bought, and the program will have to be reinstalled.

2.14 Overall test strategy

First I will test the flow of control, i.e. that the forms all link to one another and every button works. This is top down testing.
Next, I will test the input validation, i.e. whether the user input validates and blank textboxes or erroneous data produce error messages, on all of the forms. This is white box testing.
Then, I will again test the flow of control with reference to the tab order of the textboxes. This is again top down testing.
I will also test any searching that I implement, with some black box testing, and also some testing that none of the database code crashes.
My final test will be that the Word documents open with the correct data and formatting required by the user, and are able to be edited, saved, and printed.

Chapter 3

Testing

3.1 Test plan

Test series	Purpose	Forms tested
1	Flow of control tests: menus, buttons, forms linking to each other and opening when required.	All of the forms.
2	Input validation: textboxes and user input.	All of the data input forms.
3	Tab order.	All of the data input forms.
4	Search box testing.	frmList.
5	Database data manipulation: fields inserted correctly, no crashing of the program.	Data input and retreival forms.
6	Word document manipulation: test that they open and close perfectly and that all the required data is inserted in them.	frmReportQuote, frmReportLog, frmReportInvoice.

3.2 Test data

Data number	Normal	Boundary	Erroneous
1	01234567890123	N/A	[empty]
2	Hello@example.com	N/A.	Helloexample.com
3	Dung	N/A	[empty]
4	01473 847382	N/A	[empty]
5	25	-1	q2

3.3 Test details

The colour-coded key:

ERRONEOUS DATA	**BOUNDARY DATA**	**NORMAL DATA**

Please turn over for further test details. . .

Test number	Form	Purpose	Test data	Expected result	Actual result	Evidence	Pass/ Fail
1.1	frmMainMenu	Test that every button clicked opens the expected form.	N/A	N/A; N/A; The Add button opens frmAdd, the Remove button opens frmRemove, the List button opens frmList, the Report Forms button opens frmReportMenu, and the Quit button closes the program.	N/A; N/A; As expected.		Pass.
2.1	frmAddCustomer	Make sure that MPAN number input isn't NULL, i.e. that the program does not error.	Test data 1.	An error message displays if the input is NULL.; N/A; The MPAN number field has data in it, so does not crash but inserts the data if all other checks are correct.	As expected.; N/A; As expected.		Pass.
2.2	frmAddCustomer	Make sure that the email address contains an '@' symbol; if it does not, error.	Test data 2.	An error message displays if the email address textbox is blank or does not contain an at symbol; N/A; The email address contains an at symbol, so no errors display and the data is inserted into the database if all other checks are successful.	As expected.; N/A; As expected.	See screenshot 3.1 on page 35.	Pass.

Test number (cont.)	**Form (cont.)**	**Purpose (cont.)**	**Test data (cont.)**	**Expected result (cont.)**	**Actual result (cont.)**	**Evidence (cont.)**	**Pass/ Fail (cont.)**
2.3	frmAddCustomer	Make sure that neither the name nor the billing address textboxes are left blank.	Test data 3.	An error message displays if the textboxes are blank.; N/A; None of the textboxes are blank, so no errors display and the data is inserted into the database if all other checks are successful.	As expected.; N/A; As expected.		Pass.
2.4	frmAddCustomer	Make sure that the telephone number boxes are not left blank.	N/A	An error message displays if the telephone number texboxes are blank.; N/A; None of the telephone number textboxes are blank, so no errors display and the data is inserted into the database if all other checks are successful.	As expected.; N/A; As expected.		Pass.

Test number (cont.)	Form (cont.)	Purpose (cont.)	Test data (cont.)	Expected result (cont.)	Actual result (cont.)	Evidence (cont.)	Pass/ Fail (cont.)
2.5	frmAddSupplier	Make sure that the telephone number boxes are not left blank.	N/A	An error message displays if the telephone number texboxes are blank.; N/A; None of the telephone number textboxes are blank, so no errors display and the data is inserted into the database if all other checks are successful.	As expected.; N/A; As expected.		Pass.
2.6	frmAddSupplier	Make sure that neither the name nor the supplier address, nor the contact name textboxes are left blank.	Test data 3.	An error message displays if the textboxes are blank.; N/A; None of the textboxes are blank, so no errors display and the data is inserted into the database if all other checks are successful.	As expected.; N/A; As expected.		Pass.

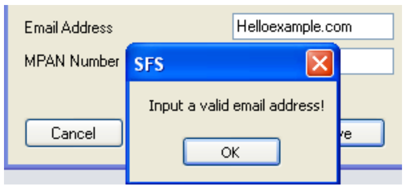
Test number (cont.)	Form (cont.)	Purpose (cont.)	Test data (cont.)	Expected result (cont.)	Actual result (cont.)	Evidence (cont.)	Pass/ Fail (cont.)
2.7	frmAddComponent	Make sure that neither the name nor the component type textboxes are left blank.	Test data 3.	An error message displays if the textboxes are blank.; N/A; None of the textboxes are blank, so no errors display and the data is inserted into the database if all other checks are successful.	As expected.; N/A; As expected.		Pass.
2.8	frmAddComponent	Make sure that only a positive numerical value is accepted in the kWp textbox.	Test data 5.	An error message displays if the textbox does not contain a number.; Error if the number entered by the user is negative.; None of the textboxes are blank and the kWp number is positive, so no errors display and the data is inserted into the database if all other checks are successful.	As expected.; N/A; As expected.	See the screenshot 3.2 on page 35.	Pass.

Test number (cont.)	Form (cont.)	Purpose (cont.)	Test data (cont.)	Expected result (cont.)	Actual result (cont.)	Evidence (cont.)	Pass/ Fail (cont.)
3.1	frmAddCustomer	Test the tab order of the customer addition form: the tab order should be logical and tabs should go from one textbox to the one straight underneath it, and from the last textbox to the Save button and then to the Cancel button.		This is simply a pass or fail test.			Pass.
3.2	frmAddSupplier	Test the tab order of the supplier addition form: the tab order should be logical and tabs should go from one textbox to the one straight underneath it, and from the last textbox to the Save button and then to the Cancel button.		This is simply a pass or fail test.			Pass.
3.3	frmAddComponent	Test the tab order of the component addition form: the tab order should be logical and tabs should go from one textbox to the one straight underneath it, and from the last textbox to the Save button and then to the Cancel button.		This is simply a pass or fail test.			Pass.

Test number (cont.)	Form (cont.)	Purpose (cont.)	Test data (cont.)	Expected result (cont.)	Actual result (cont.)	Evidence (cont.)	Pass/ Fail (cont.)
4.1	frmList	Test the search box functionality.	Test data is in the database, from the frmAddCustomer tests: testing 'banana'.	The search functionality does not search for anything.; The user entered 'Ba' and not 'Banana', so all of the results matching 'Ba' even inside the strings were returned.; The searched string displays all of its variants for the user to select.	As expected.; As expected.; As expected.	See the screenshot 3.3 on page 35.	Pass.
5.1	frmAddCustomer	Test that everything is inserted correctly into the database.	Test data used in test series 2.	Erroneous data is input into the database—unexpected blank boxes due to validation failures, for example.; N/A.; Everything is correctly inserted into the database.	As expected.; As expected.; As expected.	See screenshot 3.4 on page 36.	Pass.
6.1	frmReportQuote, frmReportLog, frmReportInvoice	Test that the Word documents open and display at the click of the relevant buttons, and that they can be edited and printed.	Previously inserted data, from the database.	The Word documents do not open; the program crashes.; N/A.; The Word documents open and can be printed and edited and saved again.	As expected.; N/A.; As expected.	See screenshots 3.5 on page 3.5.	Pass.

(For the sake of brevity, I have only included evidence of a few tests of every form.)

3.4 Associated screenshots

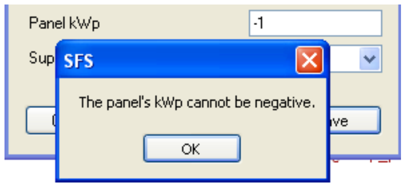

Email Address: Helloexample.com

MPAN Number

SFS

Input a valid email address!

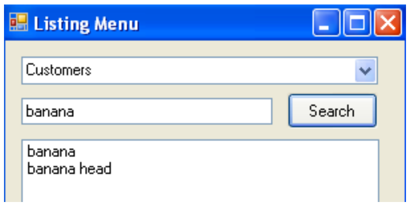
Figure 3.1: Email address error message.


Panel kWp: -1

SFS

The panel's kWp cannot be negative.

Figure 3.2: Boundary kWp data entry error.

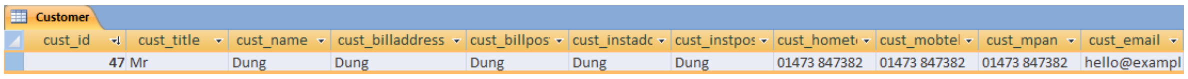

Listing Menu

Customers

banana

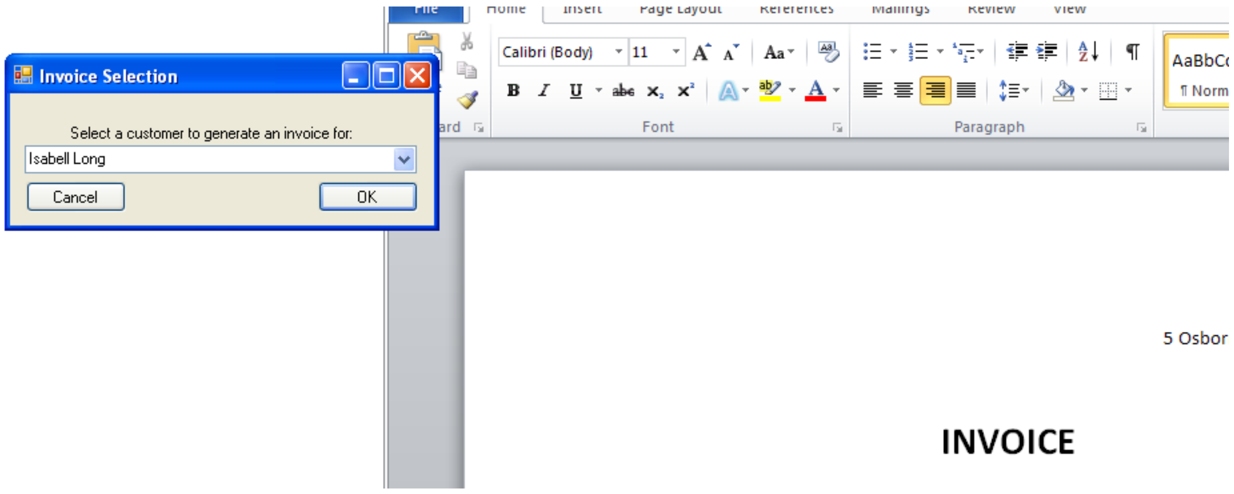
banana
banana head

Figure 3.3: The search function succeeding.


Customer

cust_id	cust_title	cust_name	cust_billaddress	cust_billpos	cust_instadc	cust_instpos	cust_homet	cust_mobtel	cust_mpan	cust_email
47	Mr	Dung	Dung	Dung	Dung	Dung	01473 847382	01473 847382	01473 847382	hello@exampl

Figure 3.4: The customer data inserted correctly into the database.


Invoice Selection

Select a customer to generate an invoice for:

Isabell Long

Cancel OK

5 Osbor

INVOICE

Figure 3.5: The invoice opening.

Chapter 4

System Maintenance

4.1 System overview

The system is written in Visual Basic.NET and consists of four major sections: the adding, deleting and listing of customers, suppliers and components into or from a database, and the production of reports.

There are nineteen forms that consist of buttons, textboxes and labels, most of which are named quite clearly. The code is split into procedures and functions, many of which control the buttons. The user enters data into or selects data from the menus in the form and then clicks whichever button he or she wants to, usually the 'OK' or 'Save' button and rarely the cancel button once all the data has been entered into the textboxes.

The database which the data is added into, removed from, or obtained from for the purposes of listing is a Microsoft Access database which is queried with runtime SQL queries and consists of four tables: Login, Customer, Supplier, and Component. The login details are static: they do not change and cannot be removed or edited from anywhere in the program, just in the database which it is assumed that the user will not want to touch. The Customer table contains the customer details, and the Supplier and Component details contain the supplier and component details. The program controls adding data to and removing and listing data from the tables, according to what the user chooses to do.

All but one of the reports, the incomplete plain Visual Basic form named 'frmSurveyForm', displays the report in Microsoft Word. The reports for logs, quotations and invoices are opened in Microsoft Word. The log form is the easiest to understand in that the user just has to select a month and a year and the respective Microsoft Word document named 'log_[month][year].docx' is opened. For the quotations and invoices, some more complicated processing happens—the word 'magic' in the WordQuotationAutomationMagic() and WordInvoiceAutomationMagic() in procedure names is testament to that! The quotations and invoices have data selected from the database using 'SELECT [required fields] FROM [specific table]' runtime SQL queries, of which the output is assigned to variables with cryptic names that only I can remember what they stand for (see the variable lists!), then input into Word paragraph objects and displayed with some formatting such as font size and centering that is also coded and not done in Word.

4.2 Samples of forms and reports

4.2.1 Forms

Name	Purpose
frmLoginForm	The login form.
frmMainMenu	The main menu form.
frmAdd	The customer/supplier/component addition menu form.
frmAddCustomer	The customer addition form.
frmAddSupplier	The supplier addition form.
frmAddComponent	The. . . oooh, take a guess. . .
frmRemove	The customer/supplier/component deletion menu form.
frmRemoveCustomer	The customer details removal form.
frmRemoveSupplier	The supplier details removal form.
frmRemoveComponent	The component details removal form.
frmList	The customer/supplier/component listing form.
frmListFullDetails	Lists the full details of the selected customer/supplier/component in a form.
frmReportMenu	the report generating/viewing menu, listing the different types of reports.
frmReportInvoice	The report invoice selection and manipulation menu—opens Word documents for invoices.
frmReportLog	The log file selection and manipulation menu—opens Word documents for log forms.
frmReportQuote	The quotation selection menu—opens Word documents for quotations.
frmReportSurvey	The survey form selection menu.
frmSurveyForm	Displays survey details of the selected customer.
frmSwankyCode	Not really a form—just a form interface for code behind it referenced by other parts of code—'swanky code' that could be pushed out so as to not clutter up other forms' code.

4.2.2 Reports

There are no reports generated by the program itself, in the program: data is just piped into Microsoft Word documents.

4.3 A sample of detailed algorithm design

Algorithm to check whether the user-entered email address contains '@', therefore if it is valid or not.

```
$pos \leftarrow 0$
$etb \leftarrow txtCustomerEmail.Text$
$atsymbol \leftarrow' @'$
$pos \leftarrow InStr(etb, atsymbol)$
if $pos = 0$ then
    $ErrorYesNo \leftarrow True$
    $intInsert \leftarrow 0$
$Box("Input\ a\ valid\ email\ address!")$
else
$ErrorYesNo \leftarrow False$
end if
```

All other algorithms defined in the Design section have stayed essentially the same.

4.4 Procedure, function and variable lists

4.4.1 Procedures and functions

4.4.1.1 frmLoginForm

Name	Type	Purpose
frmLoginForm_Load	Procedure	The automatically created procedure that houses the code executed when the form loads. In this case, it loads the database connection string.
OK_Click	Procedure	The procedure that controls what happens when the user clicks on the OK button. In this case, the username and passwords entered by the user are compared with those in the database and an error is spat out if the credentials are wrong, else the main menu is displayed.
Cancel_Click	Procedure	The code inside here executes when the user clicks the Cancel button on this form. The program closes.

4.4.1.2 frmMainMenu

Name	Type	Purpose
frmMainMenu_Load	Procedure	Controls what happens when the main menu form loads. In this case, the login form gets hidden.
btnReport_Click	Procedure	Controls what happens when the Report Forms button is clicked. The main menu is hidden and ‘frmReportMenu’ appears.
btnList_Click	Procedure	As above, but with the List button and ‘frmList’.
btnAdd_Click	Procedure	As above, but with the Add button and ‘frmAdd’.
btnRemove_Click	Procedure	As above, but with the Remove button and ‘frmRemove’.
btnQuit_Click	Procedure	When the user clicks the Quit button, the program closes. There is no point returning the user to the login form as this could cause problems with the database and the opening and closing of connections.

4.4.1.3 frmAdd

Name	Type	Purpose
frmAdd_Load	Procedure	This contains code to check if the database connection is still active, and populate the form's dropdown box, when the form loads.
btnShow_Click	Procedure	When the user clicks the Show button, the code inside this procedure makes the corresponding form display.
btnClose_Click	Procedure	Controls what happens when the Close button is clicked. The current form closes and the main menu is displayed.

4.4.1.4 frmAddCustomer

Name	Type	Purpose
frmAddCustomers_Load	Procedure	This contains code to check if the database connection is still active, and populate the form's dropdown box, when the form loads, and also control the tooltips that display when the user hovers over certain labels.
btnSave_Click	Procedure	This takes all the user-entered details and saves them to the database, or displays an error message if some don't validate.
InsertParameters	Procedure	This takes the textbox values and makes sure that each one goes into the correct database field.
btnCancel_Click	Procedure	When the user clicks the Cancel button, this executes code to hide this form and display the previous one, 'frmAdd'.

4.4.1.5 frmAddSupplier

Name	Type	Purpose
frmAddSupplier_Load	Procedure	Executes code when the form loads. In this case, it checks whether the database connection is alive.
btnSave_Click	Procedure	This takes all the user-entered details and saves them to the database, or displays an error message if some don't validate.
InsertParameters	Procedure	This takes the textbox values and makes sure that each one goes into the correct database field.
btnCancel_Click	Procedure	When the user clicks the Cancel button, this executes code to hide this form and display the previous one, 'frmAdd'.

4.4.1.6 frmAddComponent

Name	Type	Purpose
AddComponent_Load	Procedure	This contains code to check if the database connection is still active, and populate the form's dropdown box, when the form loads.
btnSave_Click	Procedure	This takes all the user-entered details and saves them to the database, or displays an error message if some don't validate.
InsertParameters	Procedure	This takes the textbox values and makes sure that each one goes into the correct database field.
btnCancel_Click	Procedure	When the user clicks the Cancel button, this executes code to hide this form and display the previous one, 'frmAdd'.

4.4.1.7 frmRemove

Name	Type	Purpose
frmRemove_Load	Procedure	This contains code to check if the database connection is still active, and populate the form's dropdown box, when the form loads.
btnShow_Click	Procedure	When the user clicks the Show button, the code inside this procedure makes the corresponding form display.
btnClose_Click	Procedure	Controls what happens when the Close button is clicked. The current form closes and the main menu is displayed.

4.4.1.8 frmRemoveCustomer

Name	Type	Purpose
frmRemoveCustomers_Load	Procedure	This contains code to check if the database connection is still active, and populate the form's dropdown box, when the form loads.
btnRemove_Click	Procedure	On click, this button's code takes the customer name that the user selected and removes all of their details from the database, or displays an error message if the deletion fails.
RemoveParameters	Procedure	This takes the user-selected value and makes sure that the correct customer is deleted from the database, by specifying which field the user-selected data corresponds to.
btnCancel_Click	Procedure	When the user clicks the Cancel button, this executes code to hide this form and display the previous one, 'frmRemove'.

4.4.1.9 frmRemoveSupplier

Name	Type	Purpose
frmRemoveSupplier_Load	Procedure	This contains code to check if the database connection is still active and populate the form's dropdown box when the form loads.
btnRemove_Click	Procedure	On click, this button's code takes the supplier name that the user selected and removes all of their details from the database, or displays an error message if the deletion fails.
RemoveParameters	Procedure	This takes the user-selected value and makes sure that the correct supplier is deleted from the database, by specifying which field the user-selected data corresponds to.
btnCancel_Click	Procedure	When the user clicks the Cancel button, this executes code to hide this form and display the previous one, 'frmRemove'.

4.4.1.10 frmRemoveComponent

Name	Type	Purpose
frmRemoveComponent_Load	Procedure	This contains code to check if the database connection is still active and populate the form's dropdown box when the form loads.
btnRemove_Click	Procedure	On click, this button's code takes the component name that the user selected and removes all of their details from the database, or displays an error message if the deletion fails.

Name (cont.)	Type (cont.)	Purpose (cont.)
RemoveParameters	Procedure	This takes the user-selected value and makes sure that the correct component is deleted from the database, by specifying which field the user-selected data corresponds to.
btnCancel_Click	Procedure	When the user clicks the Cancel button, this executes code to hide this form and display the previous one, 'frmRemove'.

4.4.1.11 frmList

Name	Type	Purpose
frmList_Load	Procedure	This contains code to check if the database connection is still active, and populate the form's dropdown box, when the form loads.
cbtxtListOptions_SelectedIndexChanged	Procedure	Lists all of the customers, suppliers or components, depending on what the user selects in this dropdown box. This lists them straight away.
btnSearch_Click	Procedure	On click, this will search for the string that the user entered into the search box, depending on what the user selected to search for in the dropdown box: customers, suppliers, or components.
btnShowAssoc_Click	Procedure	This procedure contains code that calls the AssocCompSupp() procedure if the selected dropdown item is Component. Otherwise, it displays an error message.
AssocCompSupp	Procedure	This finds out from the database in a slightly complicated way which supplier supplies the selected component, and displays it in a message box. The supplier in the database is an ID, so the following function converts it into a name and gives that value back to this procedure to print the full dialogue box in a way the user will understand.
SwitchSuppIDToName	Function	This function gets the supplier ID produced by the database code in the AssocCompSupp() procedure and, with the use of its value (passed by value), converts the ID into a name, then puts that into a variable which is originally passed by reference from AssocCompSupp() so that its value changes in that procedure to, producing a complete, user-comprehensible (hopefully!) message box saying which component is supplied by which supplier.
btnListFullDetails_Click	Procedure	When the user clicks on this button, the form that has the ability and space to list the full customer/supplier/component details displays, and this form hides.
btnClose_Click	Procedure	As usual, on click, this form is hidden and the previous form, in this case the main menu, is shown.

4.4.1.12 frmListFullDetails

Name	Type	Purpose
frmListFullDetails_Load	Procedure	This makes sure that the database connection is still alive, and contains an If statement that calls various procedures to populate the full information depending on what the user selected from the dropdown menu on the previous form.

Name (cont.)	Type (cont.)	Purpose (cont.)
PopulateCustInfo	Procedure	This populates the customer-related textboxes depending on the customer name the user has selected on the previous form, frmList.
PopulateSuppInfo	Procedure	This does the same, but for suppliers and supplier information.
PopulateCompInfo	Procedure	This does the same, but for components and component information.
btnClose_Click	Procedure	This form is closed, and 'frmList' is unhidden.

4.4.1.13 frmReportMenu

Name	Type	Purpose
btnLogs_Click	Procedure	On click, display 'frmReportLog' and hide this form.
btnQuotes_Click	Procedure	On click, display 'frmReportQuotes' and hide this form.
btnInvoices_Click	Procedure	On click, display 'frmReportInvoice' and hide this form.
btnSurvey_Click	Procedure	On click, display 'frmReportSurvey' and hide this form.
btnQuit_Click	Procedure	This form is closed, and 'frmMainMenu' is unhidden.

4.4.1.14 frmReportInvoice

Name	Type	Purpose
frmReportInvoice_Load	Procedure	This checks if the database connection is still alive, then gets the customer name from the database to populate the dropdown menu from which the user selects a customer name.
btnOK2_Click	Procedure	When the user clicks OK, code is executed that gets all the relevant customer details from the database and puts each one into a variable.
btnCancel_Click	Procedure	This form is closed, and 'frmReportMenu' is unhidden.

4.4.1.15 frmReportLog

Name	Type	Purpose
frmReportLog_Load	Procedure	When the form loads, code is executed that populates the dropdown list with months and years that the user can select from to view a log file.
btnOK_Click	Procedure	When the user clicks OK, the log form corresponding to the month and year the user selected is opened in Microsoft Word.
btnCancel_Click	Procedure	This form is closed, and 'frmReportMenu' is unhidden.

4.4.1.16 frmReportQuote

Name	Type	Purpose
frmReportQuote_Load	Procedure	This checks if the database connection is still alive, then gets the customer name from the database to populate the dropdown menu from which the user selects a customer name.

Name (cont.)	Type (cont.)	Purpose (cont.)
btnOK2_Click	Procedure	When the user clicks OK, code is executed that gets all the relevant customer details from the database and puts each one into a variable.
btnCancel_Click	Procedure	This form is closed, and ‘frmReportMenu’ is unhidden.

4.4.1.17 frmReportSurvey

Name	Type	Purpose
frmReportSurvey_Load	Procedure	This checks if the database connection is still alive, then gets the customer name from the database to populate the drop-down menu from which the user selects a customer name.
btnOK_Click	Procedure	When the user clicks OK, the user’s selected item in the dropdown list is attributed to a variable, and ‘frmSurveyForm’ is shown.
btnCancel_Click	Procedure	This form is closed, and ‘frmReportMenu’ is unhidden.

4.4.1.18 frmSurveyForm

Name	Type	Purpose
frmSurveyForm_Load	Procedure	The load procedure that executes the obtaining of specific customer details when the form loads.
btnClose_Click	Procedure	When this button is clicked, code executes that closes this form and opens ‘frmReportSurvey’.

4.4.1.19 frmSwankyCode

Name	Type	Purpose
CheckAdditions	Procedure	This is referenced by other code in other forms to check the additions into textboxes to make sure that everything goes into the database correctly.
WordInvoiceAutomationMagic	Procedure	This is referenced from the invoice form. This procedure deals with inserting the correct values in the correct place in the invoice Word document, and opening it to the user.
WordQuotationAutomationMagic	Procedure	This is referenced from the quotation form. This procedure deals with inserting the correct values in the correct place in the quotation Word document, and opening it to the user.

4.4.2 Variables

4.4.2.1 frmLoginForm

Variable	Data Type	Purpose
accConnection	OleDbConnection	The database connection’s main command to make Access work with VB.

Variable (cont.)	Data Type (cont.)	Purpose (cont.)
ConnectionString	String	The variable that defines the database provider and the location of the database.
cmdString	String	The variable for the SQL statement that is passed to the database.
da	OleDbDataAdapter	The dataadapter variable, which takes cmdString and accConnection as parameters.
ds	DataSet	The dataset variable.
dt	DataTable	The datatable variable which allows tables in the database to be referenced.
lblUsername	Label	The username form label, to tell the user to enter a username in the box below it.
lblPassword	Label	The password form label, to tell the user to enter a password in the box below it.
txtUsername	Textbox	The textbox that the user enters their username into on the form.
txtPassword	Textbox	As above, but the user enters a password into it.
btnCancel	Button	The cancel button on this form.
btnOK	Button	The OK button—on click, this executes the code within the btnOK_Click procedure, checking the user-entered values with the ones in the database.
u	String	This takes the value of txtUsername's .Text property for ease of typing later on when doing the comparison.
p	String	The same as above, but the value of txtPassword's .Text property.
loginuname	String	The variable for the actual value of the database 'login_uname' field.
loginpword	String	The variable for the actual value of the database 'login_pword' field.

4.4.2.2 frmMainMenu

Variable	Data Type	Purpose
btnAdd	Button	On click, this button opens the addition menu.
btnRemove	Button	On click, this button opens the removal menu.
btnList	Button	On click, this button opens the listing form.
btnReport	Button	On click, this button displays the report menu.
btnQuit	Button	On click, this closes the program.

4.4.2.3 frmAdd

Variable	Data Type	Purpose
cbtxtAddOptions	ComboBox	The dropdown menu/textbox for the addition options: customer/ supplier/component.
btnShow	Button	The button that shows the next form, depending on what the user selects in the dropdown menu.
btnClose	Button	On click, this button closes frmAdd and opens the previous form.
accConnection	OleDbConnection	This is not a variable directly in this form: it is referenced from frmLoginForm where it has values.

4.4.2.4 frmAddCustomer

Variable	Data Type	Purpose
lblCustomerTitle	Label	The customer title label, to tell the user to enter a customer's title in the box beside it.
lblCustomerName	Label	As above, but the customer name label.
lblCustomerBillingAddress	Label	As above, but the customer billing address label.
lblCustomerBillingPostcode	Label	As above, but the customer billing postcode label.
lblCustomerInstallationAddress	Label	As above, but the customer installation address label.
lblCustomerInstallationPostCode	Label	As above, but the customer installation postcode label.
cbCustomerBillInstallSameAddress	Checkbox	The user ticks this box if the customer's installation address is the same as his or hers billing address.
lblCustomerHomeTelNo	Label	The customer home telephone number label, to tell the user to enter a customer's home telephone number into the box beside it.
lblCustomerMobTelNo	Label	As above, but with the customer's mobile telephone number.
lblCustomerEmail	Label	As above, but the email address of the customer.
lblCustomerMpanNo	Label	As above, but the customer's MPAN number.
cbCustomerTitle	Combobox	The actual combobox. Using this, the user can specify a title for the specific customer, for example Mr or Mrs.
txtCustomerName	Textbox	The user can enter the customer name into this textbox.
txtCustomerBillAddress	Textbox	The user can enter the customer's billing address into this textbox.
txtCustomerBillPostcode	Textbox	As above, but the customer's billing address postcode.
txtCustomerHomeTelNo	Textbox	As above, but the customer's home telephone number.
txtCustomerMobTelNo	Textbox	As above, but the customer's mobile telephone number.
txtCustomerEmail	Textbox	As above, but the customer's email address.
txtCustomerMpanNo	Textbox	As above, but the customer's MPAN number.
btnCancel	Button	This is the name of the button on this form that, when clicked, will hide the current form and display 'frmAdd'.
btnSave	Button	This button, when clicked, will execute code that will save the data the user has entered to the database, or error if there is a problem.
accConnection	OleDbConnection	This is referenced in the _Load procedure from frmLoginForm where it has values, but is also a variable in its own right so that the code in the _Click procedure can execute (database update stuff).
cmdString	String	The variable that holds the SQL command string for INSERT INTO to make the customer details go into the database.
AddCustomerDataAdapter	OleDbDataAdapter	The dataadapter variable: it takes the accConnection variable value that doesn't change, and the strSQL output. Well, in this case it does nothing. . . .
accCommand	OleDbCommand	The accCommand variable renamed so as not to confuse the compiler when passed by reference to the InsertParameters() procedure in order to give the database fields values based on values in the corresponding textboxes.
intInsert	Integer	This is the checking variable—if it its value is ever 0 after a database operation, the operation has not completed successfully.
acccmd	OleDbCommand	The accCommand variable renamed so as not to confuse the compiler when passed by reference to the InsertParameters() procedure in order to give the database fields values based on values in the corresponding textboxes.

Variable (cont.)	Data Type (cont.)	Purpose (cont.)
pos	Integer	This is the positioning variable that is used in the InStr() string manipulation code that finds the position of an ‘’ symbol.
etb	String	The shortened name for txtCustomerEmail’s .Text property.
atsymbol	String	The atsymbol, to avoid having a symbol directly in the InStr() operation.

4.4.2.5 frmAddSupplier

Variable	Data Type	Purpose
lblSupplierName	Label	The supplier name label, to tell the user to enter a supplier name in the box beside it.
lblSupplierAddress	Label	The supplier address label, to tell the user to enter a supplier address in the box beside it.
lblSupplierPostCode	Label	The supplier postcode label, to tell the user to enter a supplier postcode in the box beside it.
lblSupplierTelNo	Label	The supplier telephone number label, to tell the user to enter a supplier telephone number in the box beside it.
lblSupplierContactName	Label	The supplier contact name label, to tell the user to enter a supplier contact name in the box beside it.
txtSupplierName	Textbox	The textbox that the user enters the supplier name into on the form.
txtSupplierAddress	Textbox	As above, but the user enters the supplier’s address.
txtSupplierPostCode	Textbox	As above, but the user enters the supplier’s postcode.
txtSupplierTelNo	Textbox	As above, but the user enters the supplier’s telephone number.
txtSupplierContactName	Textbox	As above, but the user enters the name of the supplier’s employee which they have the most contact with and/or who is their designated contact person.
btnCancel	Button	Closes frmAddSupplier and reopens frmAdd.
btnSave	Button	This executes code that saves the supplier details, erroring if there is a problem.
accConnection	OleDbConnection	This is referenced in the _Load procedure from frmLoginForm where it has values, but is also a variable in its own right so that the code in the _Click procedure can execute (database update stuff).
cmdString	String	This variable takes the INSERT INTO Supplier SQL query, in this case.
AddSupplierDataAdapter	OleDbDataAdapter	This is a variable defining the DataAdapter
accCommand	OleDbCommand	This is the variable which allows properties of it to be set such as the command text defined as the value of cmdString, and the location of the accConnection (frmLoginForm, as mentioned earlier).
intInsert	Integer	This is the checking variable—if it its value is ever 0 after a database operation, the operation has not completed successfully.
acccmd	OleDbCommand	The accCommand variable renamed so as not to confuse the compiler when passed by reference to the InsertParameters() procedure in order to give the database fields values based on values in the corresponding textboxes.

4.4.2.6 frmAddComponent

Variable	**Data Type**	**Purpose**
btnCancel	Button	The cancel button, which, when clicked, executes code that closes this form and shows 'frmAdd' again.
btnSave	Button	On click, the data entered is taken and saved into the database, or an error message pops up if something is wrong with the data.
lblComponentName	Label	The component name label, to tell the user to enter a component name in the box beside it.
lblComponentType	Label	The component type label, to tell the user to enter a component type in the box beside it.
lblComponentSerialNo	Label	The component serial number label, to tell the user to enter a component serial number in the box beside it.
lblComponentPanelkWp	Label	The component panel kWp label, to tell the user to enter the kWp of the solar panel in the box beside it.
lblCompSupplierName	Label	The component supplier name label, to tell the user to enter the supplier name in the box beside it.
txtComponentName	Textbox	The textbox that the user enters the component name into on the form.
txtComponentType	Textbox	As above, but the user enters the component type into the form.
txtComponentSerialNo	Textbox	As above, but the user enters the component serial number into the form.
txtComponentPanelkWp	Textbox	As above, but the user enters the solar panel kWp value into the form.
txtCompSupplierName	Textbox	As above, but the user enters the name of the supplier into the form.
accConnection	OleDbConnection	This is referenced in the _Load procedure from frmLoginForm where it has values, but is also a variable in its own right so that the code in the _Click procedure can execute (database update stuff).
strSQL	String	This variable takes the INSERT INTO Component SQL query, in this case.
da	OleDbDataAdapter	The dataadapter variable: it takes the accConnection variable value that doesn't change, and the strSQL output.
ds	DataSet	The dataset variable.
dt	DataTable	The datatable variable which allows tables in the database to be referenced.
dr	DataRow	The datarow variable.
suppidvaluecmd	String	This takes the SELECT SQL command string that gets the supplier ID value from the Supplier table, when the supplier name = the selected supplier name selected by the drop down box by the user.
cmdString	String	This is the INSERT INTO SQL string that inserts the values entered by the user (in the form) into the database, including the now enterable value of the comp_supplier field, the ID, through some data fiddling.
da	OleDbDataAdapter	The dataadapter variable: it takes the accConnection variable value that doesn't change, and the suppidvaluecmd output.
ds	DataSet	The dataset variable.
accCommand	OleDbCommand	This is the variable which allows properties of it to be set such as the command text defined as the value of cmdString, and the location of the accConnection (frmLoginForm, as mentioned earlier).

Variable (cont.)	**Data Type (cont.)**	**Purpose (cont.)**
intInsert	Integer	This is the checking variable—if it its value is ever 0 after a database operation, the operation has not completed successfully.
supplieridvalue	Integer	The supplier ID value from the database, into a variable, so it can be put into the comp_supplier database field in the Component table later on.

4.4.2.7 frmRemove

Variable	**Data Type**	**Purpose**
cbtxtRemoveOptions	ComboBox	The dropdown menu/textbox for the removal options: customer/ supplier/component.
btnShow	Button	The button that shows the next form, depending on what the user selects in the dropdown menu.
btnClose	Button	On click, this button executes code that closes frmAdd and opens the previous form.
accConnection	OleDbConnection	This is not a variable directly in this form: it is referenced from frmLoginForm where it has values.

4.4.2.8 frmRemoveCustomer

Variable	**Data Type**	**Purpose**
btnCancel	Button	When clicked, code behind this button closes the current form and displays ‘frmRemove’.
btnRemove	Button	This button, when clicked, executes code that removes the selected customer from the database.
txtcbCustomerRemove	Combobox	This is the dropdown box from which the user can select a customer to remove.
strSQL	String	This variable takes the DELETE FROM Customer table SQL query, in this case.
da	OleDbDataAdapter	The dataadapter variable: it takes the accConnection variable value that doesn’t change, and the strSQL output.
ds	DataSet	The dataset variable.
dt	DataTable	The datatable variable which allows tables in the database to be referenced.
dr	DataRow	This helps in the loop to populate the dropdown box to get the customers for the user to choose which one to delete: it is used in the For Each loop to get the customer name for every customer in each row of the selected table.
cmdString	String	In the btnRemove_Click procedure, this is the string variable that holds the SQL for the DELETE of a specific customer.
da	OleDbDataAdapter	The dataadapter variable: it takes the accConnection variable value that doesn’t change, and the cmdString output.
ds	DataSet	The dataset variable.
accCommand	OleDbCommand	This is the variable which allows properties of it to be set such as the command text defined as the value of cmdString, and the location of the accConnection (frmLoginForm, as mentioned earlier).

Variable (cont.)	Data Type (cont.)	Purpose (cont.)
intRemove	Integer	This is the checking variable—if it its value is ever 0 after a database operation, the operation has not completed successfully.
acccmd	OleDbCommand	The accCommand variable renamed so as not to confuse the compiler when passed by reference to the InsertParameters() procedure in order to give the database fields values based on values in the corresponding textboxes.

4.4.2.9 frmRemoveSupplier

Variable	Data Type	Purpose
btnCancel	Button	When clicked, code behind this button closes the current form and displays ‘frmRemove’.
btnRemove	Button	This button, when clicked, executes code that removes the selected supplier from the database.
txtcbSupplierRemove	Combobox	This is the dropdown box from which the user can select a supplier to remove.
da	OleDbDataAdapter	The dataadapter variable: it takes the accConnection variable value that doesn't change, and the strSQL output.
ds	DataSet	The dataset variable.
dt	DataTable	The datatable variable which allows tables in the database to be referenced.
dr	DataRow	This helps in the loop to populate the dropdown box to get the suppliers for the user to choose which one to delete: it is used in the For Each loop to get the supplier name for every supplier in each row of the selected table.
cmdString	String	In the btnRemove_Click procedure, this is the string variable that holds the SQL for the DELETE of a specific supplier.
da	OleDbDataAdapter	The dataadapter variable: it takes the accConnection variable value that doesn't change, and the cmdString output.
ds	DataSet	The dataset variable.
accCommand	OleDbCommand	This is the variable which allows properties of it to be set such as the command text defined as the value of cmdString, and the location of the accConnection (frmLoginForm, as mentioned earlier).
intRemove	Integer	This is the checking variable—if it its value is ever 0 after a database operation, the operation has not completed successfully.
acccmd	OleDbCommand	The accCommand variable renamed so as not to confuse the compiler when passed by reference to the InsertParameters() procedure in order to give the database fields values based on values in the corresponding textboxes.

4.4.2.10 frmRemoveComponent

Variable	Data Type	Purpose
btnCancel	Button	When clicked, code behind this button closes the current form and displays ‘frmRemove’.
btnRemove	Button	This button, when clicked, executes code that removes the selected component from the database.

Variable (cont.)	**Data Type (cont.)**	**Purpose (cont.)**
txtcbComponentRemove	Combobox	This is the dropdown box from which the user can select a component to remove.
da	OleDbDataAdapter	The dataadapter variable: it takes the accConnection variable value that doesn't change, and the strSQL output.
ds	DataSet	The dataset variable.
dt	DataTable	The datatable variable which allows tables in the database to be referenced.
dr	DataRow	This helps in the loop to populate the dropdown box to get the components for the user to choose which one to delete: it is used in the For Each loop to get the component name for every component in each row of the selected table.
cmdString	String	In the btnRemove_Click procedure, this is the string variable that holds the SQL for the DELETE of a specific component.
da	OleDbDataAdapter	The dataadapter variable: it takes the accConnection variable value that doesn't change, and the cmdString output.
ds	DataSet	The dataset variable.
accCommand	OleDbCommand	This is the variable which allows properties of it to be set such as the command text defined as the value of cmdString, and the location of the accConnection (frmLoginForm, as mentioned earlier).
intRemove	Integer	This is the checking variable—if it its value is ever 0 after a database operation, the operation has not completed successfully.
acccmd	OleDbCommand	The accCommand variable renamed so as not to confuse the compiler when passed by reference to the InsertParameters() procedure in order to give the database fields values based on values in the corresponding textboxes.

4.4.2.11 frmList

Variable	**Data Type**	**Purpose**
accConnection	OleDbConnection	This is referenced in the _Load procedure from frmLoginForm where it has values, but is also a variable in its own right so that the code in the _Click procedure can execute (database update stuff).
btnClose	Button	On click, this button executes code that closes frmList and opens the previous form.
btnShow	Button	On click, this button executes code which populates the listbox below it with the values of whatever the user selected in the dropdown menu.
lbList	ListBox	The box that holds the output of the command executed by the code behind the btnShow button—the list of customer/suppliers/ components.
cbtxtListOptions	ComboBox	The dropdown menu/textbox for the listing options: customer/ supplier/component.
btnShowAssoc	Button	This executes the code that shows the associations between the different components and suppliers, for example.
btnListFullDetails	Button	When this button is clicked, it executes code that opens the frmListFullDetails form, listing the full details of the selected customer (from lbList) in the appropriate boxes.

Variable (cont.)	Data Type (cont.)	Purpose (cont.)
cmdString	String	This is the SQL command string that changes depending on what is selected and what needs to be displayed in lbList to pull the correct data from the database.
ds	DataSet	The dataset variable in the main procedure.
dr	DataSet	The datarow variable, which gets the current row of data based on the cmdString, and outputs it in lbList.Items.Add(), in this case.
da0	OleDbDataAdapter	The dataadapter variable for if the user selects customers: it takes the accConnection variable value that doesn't change, and the Customer table cmdString.
da1	OleDbDataAdapter	The dataadapter variable for if the user selects suppliers: it takes the accConnection variable value that doesn't change, and the Supplier table cmdString.
da2	OleDbAdapter	The dataadapter variable for if the user selects components: it takes the accConnection variable value that doesn't change, and the Component table cmdString.
dt0	DataTable	The datatable variable which allows tables in the database to be referenced based in this case on the Customer table's cmdString.
dt1	DataTable	The datatable variable which allows tables in the database to be referenced based in this case on the Supplier table's cmdString.
dt2	DataTable	The datatable variable which allows tables in the database to be referenced based in this case on the Component table's cmdString.
cn	String	In the AssocCompSupp() (association component/supplier) procedure, this takes the selected component name from the lbList and assigns it to cn.
compsuppliervalcmd	String	This is the SQL command that gets the component's supplier ID from the Component table (it's a foreign key in that table) according to the component name that is 'cn'.
sn	String	This is the supplier name variable—this will be filled in by the value returned by the operations in the SwitchSuppIDToName() function.
da	OleDbDataAdapter	The dataadapter variable, which takes compsuppliervalcmd and accConnection as parameters.
ds	DataSet	The dataset variable.
compsupplierval	Integer	This gets the supplier ID of the component supplier from the data retrieved from the SQL command.
csuppv	Integer	This is assigned the value of the above compsupplierval variable, purely for ease of use in the function that converts the supplier ID to the name of the supplier, SwitchSuppIDToName().
csuppval	Integer	The csuppv variable from the AssocCompSupp() procedure, passed by value to the SwitchSuppIDToName() function, given a different name to avoid confusing the compiler.
finalsn	String	The sn variable from the AssocCompSupp() procedure, passed by reference to the SwitchSuppIDToName() function, given a different name to again avoid confusing the compiler. Later on, this is given the value of the result of the cmdString in this function.
cmdString	String	The variable that is the SQL statement that gets the supplier's name from the Supplier table, where the supplier ID is equal to the variable csuppval.

Variable (cont.)	Data Type (cont.)	Purpose (cont.)
da	OleDbDataAdapter	The dataadapter variable: it takes the accConnection variable value that doesn't change, and the cmdString output.
ds	DataSet	The dataset variable.

4.4.2.12 frmListFullDetails

Variable	Data Type	Purpose
lblFullListCustomer	Label	The label that tells the user what the particular section of this form is showing. In this case, the customer details.
lblFullListSupplier	Label	The label that tells the user what the particular section of this form is showing. In this case, the supplier details.
lblFullListComponent	Label	The label that tells the user what the particular section of this form is showing. In this case, the component details.
btnClose	Button	On click, this form closes and the previous form, 'frmList', shows itself.
lblCustomerTitle	Label	The customer title label, to tell the user that the customer's title is listed in the box beside it.
lblCustomerName	Label	The customer name label, to tell the user that the customer's name is listed in the box beside it.
lblCustomerBillingAddress	Label	The customer billing address label, to tell the user that the customer's billing address is listed in the box beside it.
lblCustomerBillingPostcode	Label	The customer billing postcode label, to tell the user that the customer's billing postcode is listed in the box beside it.
lblCustomerInstallationAddress	Label	The customer installation address label, to tell the user that the customer's installation address is listed in the box beside it.
lblCustomerInstallationPostCode	Label	The customer installation postcode label, to tell the user that the customer's installation postcode is listed in the box beside it.
lblCustomerHomeTelNo	Label	The customer home telephone number label, to tell the user that the customer's home telephone number is listed in the box beside it.
lblCustomerMobTelNo	Label	The customer mobile telephone number label, to tell the user that the customer's mobile telephone number is listed in the box beside it.
lblCustomerEmail	Label	The customer email address label, to tell the user that the customer's email address is listed in the box beside it.
lblCustomerMpanNo	Label	The customer MPAN number label, to tell the user that the customer's MPAN number is listed in the box beside it.
lblSupplierName	Label	The supplier name label, to tell the user that the supplier's name is what is listed in the box beside it.
lblSupplierAddress	Label	The supplier address label, to tell the user that the supplier's address is what is listed in the box beside it.
lblSupplierPostCode	Label	The supplier postcode label, to tell the user that the supplier's postcode is listed in the box beside it.
lblSupplierTelNo	Label	The supplier telephone number label, to tell the user that the supplier's telephone number is listed in the box beside it.
lblSupplierContactName	Label	The supplier contact name label, to tell the user that the supplier's telephone number is listed in the box beside it.

Variable (cont.)	**Data Type (cont.)**	**Purpose (cont.)**
lblComponentName	Label	The component name label, to tell the user that the component name is listed in the box beside it.
lblComponentType	Label	The component type label, to tell the user that the component type is listed in the box beside it.
lblComponentSerialNo	Label	The component serial number label, to tell the user that the component serial number is listed in the box beside it.
lblComponentPanelkWp	Label	The component panel kWp label, to tell the user that the kWp of the solar panel is listed in the box beside it.
lblCompSupplierName	Label	The component supplier name label, to tell the user that the supplier name is listed in the box beside it.
cbldctt	Textbox	These variables are of the format *type, 'list details', 'what it lists'*, to make you aware. So this one is "combobox, list details, customer title". The title of the selected customer is obtained from the database and shown in here.
txtldctn	Textbox	The selected customer's name is shown in this textbox, according to whichever customer the user selected to see details of.
txtldctba	Textbox	The billing address of the selected customer is shown in here.
txtldctbp	Textbox	The billing postcode of the selected customer is shown in here.
txtldctia	Textbox	The installation address of the selected customer, even if it is the same as the billing address, is shown in this textbox.
txtldctip	Textbox	The customer's installation postcode is shown in this textbox, even if it is the same as the billing postcode.
txtldctea	Textbox	The customer's email address is shown in this textbox.
txtldctmpan	Textbox	The customer's MPAN number is shown in this textbox.
txtldspn	Textbox	These variables are of the format *type, 'list details', 'what it lists'*, to make you aware. So this one is "combobox, list details, supplier name". This textbox shows the supplier's name.
txtldspa	Textbox	The textbox that holds the supplier's address.
txtldspp	Textbox	The textbox that holds the supplier's postcode.
txtldsptn	Textbox	The textbox that holds the supplier's telephone number.
txtldspc	Textbox	The textbox that holds the name of the contact at the supplier.
txtldcpn	Textbox	These variables are of the format *type, 'list details', 'what it lists'*, to make you aware. So this one is "combobox, list details, component name".
txtldcpt	Textbox	The textbox that holds information about the type of component selected.
txtldcpsn	Textbox	The textbox that holds information about the component's serial number.
txtldcppkwp	Textbox	The textbox that holds information regarding the panel's kWp capacity.
txtldcpspn	Textbox	This holds the name of the supplier, after it has been complicatedly passed through a few functions in order to obtain it from the database.
accConnection	OleDbConnection	This is referenced in the _Load procedure from frmLoginForm where it has values, but is also a variable in its own right so that the code in the _Click procedure or in other procedures that reference it can execute (database update stuff).

Variable (cont.)	Data Type (cont.)	Purpose (cont.)
ctn	String	This takes the value of the frmList.lbList.SelectedItem, the customer name when the selected listing is customers, in order to work with a shorter variable name.
spn	String	This takes the value of the frmList.lbList.SelectedItem, the supplier name when the selected listing is suppliers, in order to work with a shorter variable name.
cpn	String	This takes the value of the frmList.lbList.SelectedItem, the component name when the selected listing is components, in order to work with a shorter variable name.
cmdString	String	In every procedure, whether related to customers, suppliers, or components, this is the string variable that holds the SQL for the SELECT of a specific component.
da	OleDbDataAdapter	
ds	DataSet	The dataset variable.
dt	DataTable	The datatable variable which allows tables in the database to be referenced.
compsupplierval	Integer	This gets the supplier ID of the component supplier from the data retrieved from the SQL command.
csuppv	Integer	This is equal to the compsupplierval variable, but passed by value to frmList's SwitchSuppIDToName() function, given a different name to avoid confusing the compiler.
sn	String	The supplier name, initially blank for it to be passed to frmList's SwitchSupplierIDToName() function.

4.4.2.13 frmReportMenu

Variable	Data Type	Purpose
btnSurvey	Button	When clicked, this button executes code that opens the survey form menu.
btnQuotes	Button	When clicked, this button executes code that opens the quote form menu.
btnInvoice	Button	When clicked, this button executes code that opens the invoice form menu.
btnLog	Button	When clicked, this button executes code that opens the log form menu.
btnQuit	Button	When clicked, this closes the current form and displays the previous form, in this case the main menu.

4.4.2.14 frmReportInvoice

Variable	Data Type	Purpose
lblInvoiceSelectionSpiel	Label	This is the label that tells the user what the dropdown box below it is for.
txtcbCustNameForInvoice	Combobox	This is the combobox from which the user can select a customer name for the invoice.
btnOK2	Button	The OK button for this to display the quotation in Word for the selected customer.

Variable (cont.)	**Data Type (cont.)**	**Purpose (cont.)**
btnCancel	Button	The cancel button. On click, this hides the current form and shows 'frmReportMenu'.
accConnection	OleDbConnection	This is referenced in the _Load procedure from frmLoginForm where it has values, but is also a variable in its own right so that the code in the _Click procedure can execute (database update stuff).
strSQL	String	This variable takes the SELECT FROM Customer SQL query, in this case.
da	OleDbDataAdapter	The dataadapter variable: it takes the accConnection variable value that doesn't change, and the strSQL output.
ds	DataSet	The dataset variable.
dt	DataTable	The datatable variable which allows tables in the database to be referenced.
dr	DataRow	This helps in the loop to populate the dropdown box to get the customers for the invoice: it is used in the For Each loop to get the customer name for every customer in each row of the selected table.
ctn	String	This is the short variable to house the long txtcbCustNameForInvoice.SelectedItem, i.e. what the user selects from the drop down box.
cmdString	String	This holds the SQL that gets all of the customer details from the Customer table, according to which customer is specified in 'ctn'.
da	OleDbDataAdapter	The dataadapter variable: it takes the accConnection variable value that doesn't change, and the cmdString output.
ds	DataSet	The dataset variable.
dt	DataTable	The datatable variable which allows tables in the database to be referenced.
aict	String	The variable that holds the customer title obtained from the database. This variable's abbreviation goes 'add invoice cust title', and every other one is similar.
aicn	String	As above, but the customer name.
aica	String	As above, but the customer address.
aicp	String	As above, but the customer postcode.

4.4.2.15 frmReportLog

Variable	**Data Type**	**Purpose**
ms_word	Word.Application	This sets up the Word application stuff so that all the Word manipulation and automation works.
worddoc	Word.Document	This defines the Word document, again for manipulation and automation stuff.
rlsi	String	This is the shortened version of txtcbReportLog.SelectedItem, to avoid too much typing of it.
filename	String	This holds the ending filename for the log form, after string manipulation, depending on which month and year the user selected in the dropdown box.

4.4.2.16 frmReportQuote

Variable	Data Type	Purpose
txtcbCustNameForQuotation	Combobox	The combobox from which the user can select a customer to generate a quotation for.
btnOK2	Button	The OK button for this to display the quotation in Word for the selected customer.
btnCancel	Button	The cancel button. On click, this form closes and 'frmReportMenu' reopens.
accConnection	OleDbConnection	This is referenced in the _Load procedure from frmLoginForm where it has values, but is also a variable in its own right so that the code in the _Click procedure can execute (database update stuff).
strSQL	String	This variable takes the SELECT FROM Customer SQL query, in this case.
da	OleDbDataAdapter	The dataadapter variable: it takes the accConnection variable value that doesn't change, and the strSQL output.
ds	DataSet	The dataset variable.
dt	DataTable	The datatable variable which allows tables in the database to be referenced.
dr	DataRow	This helps in the loop to populate the dropdown box to get the customers for the quotation: it is used in the For Each loop to get the customer name for every customer in each row of the selected table.
ctn	String	This is the short variable to house the long txtcbCustNameForQuotation.SelectedItem, i.e. what the user selects from the drop down box.
cmdString	String	This holds the SQL that gets all of the customer details from the Customer table, according to which customer is specified in 'ctn'.
da	OleDbDataAdapter	The dataadapter variable: it takes the accConnection variable value that doesn't change, and the cmdString output.
ds	DataSet	The dataset variable.
dt	DataTable	The datatable variable which allows tables in the database to be referenced.
aqct	String	The variable that holds the customer title obtained from the database. This variable's abbreviation goes 'add quotation cust title', and every other one is similar.
aqcn	String	As above, but the customer name.
aqca	String	As above, but the customer address.
aqcp	String	As above, but the customer postcode.

4.4.2.17 frmReportSurvey

Variable	Data Type	Purpose
txtcbReportSurvey	Combobox	The dropdown box/textbox that the user can select a customer to view the survey of from.
btnCancel	Button	The cancel button which, when clicked, hides this form and displays 'frmReportMenu'.

Variable (cont.)	Data Type (cont.)	Purpose (cont.)
btnOK	Button	When clicked, this opens the survey form with the selected customer's survey details.
cmdString	String	The SQL command that in this case selects a customer name from the Customer tables.
accConnection	OleDbConnection	This is referenced in the _Load procedure from frmLoginForm where it has values, but is also a variable in its own right so that the code in the _Click procedure can execute (database update stuff).
da	OleDbDataAdapter	The dataadapter variable: it takes the accConnection variable value that doesn't change, and the cmdString output.
ds	DataSet	The dataset variable.
dt	DataTable	The datatable variable which allows tables in the database to be referenced.
custname	String	This is the shortened version of txtcbReportSurvey.SelectedItem that is referenced in 'frmSurveyForm'.

4.4.2.18 frmSurveyForm

Variable	Data Type	Purpose
btnClose	Button	On click, this button closes the current form and displays 'frmReportSurvey'.
lblSurveyCustomerTitle	Label	The label that shows the user what is displayed in the box next to it. In this case, the customer title.
lblSurveyCustName	Label	The label that shows the user what is displayed in the box next to it. In this case, the customer name.
lblSurveyCustBillAddress	Label	The label that shows the user what is displayed in the box next to it. In this case, the customer billing address.
lblSurveyCustBillPostcode	Label	The label that shows the user what is displayed in the box next to it. In this case, the customer billing postcode.
lblSurveyCustInstAddress	Label	The label that shows the user what is displayed in the box next to it. In this case, the customer installation address.
lblSurveyCustInstPostcode	Label	The label that shows the user what is displayed in the box next to it. In this case, the customer installation postcode.
lblCustomerHomeTelNo	Label	The label that shows the user what is displayed in the box next to it. In this case, the customer home telephone number.
lblCustomerMobTelNo	Label	The label that shows the user what is displayed in the box next to it. In this case, the customer mobile number.
lblCustomerEmail	Label	The label that shows the user what is displayed in the box next to it. In this case, the customer email address.
lblCustomerMpanNo	Label	The label that shows the user what is displayed in the box next to it. In this case, the customer MPAN number.
txtscn	Textbox	The textbox for the customer name value.
cbscba	Combobox	The combobox for the customer title.
txtscba	Textbox	The textbox for the customer billing address.
txtscbp	Textbox	The textbox for the customer billing postcode.
txtscia	Textbox	The textbox for the customer installation address.
txtscip	Textbox	The textbox for the customer installation postcode.
txtschtn	Textbox	The textbox for the customer's home telephone number.

Variable (cont.)	Data Type (cont.)	Purpose (cont.)
txtscmtn	Textbox	The textbox for the customer's mobile telephone number.
txtscmpan	Textbox	The textbox for the customer's MPAN number.
txtscea	Textbox	The textbox housing the customer's email address.
cn	String	The customer name string variable that is a shortened version of frmReportSurvey.txtcbReportSurvey.SelectedItem.
cmdString	String	The SQL statement that controls the SELECT to get customer details from the Customer table according to the customer the user selected.
accCommand	OleDbCommand	This is the variable which allows properties of it to be set such as the command text defined as the value of cmdString, and the location of the accConnection (frmLoginForm, as mentioned earlier).
da	OleDbDataAdapter	The dataadapter variable: it takes the accConnection variable value that doesn't change, and the cmdString output.
ds	DataSet	The dataset variable.
dt	DataTable	The datatable variable which allows tables in the database to be referenced.

4.4.2.19 frmSwankyCode

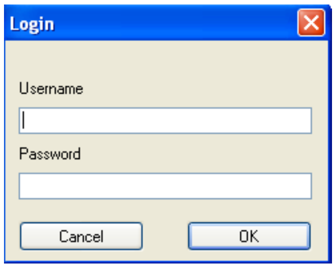
Variable	Data Type	Purpose
intInsert	Integer	This is referenced by value to the CheckAdditions() procedure: this is used to check if the insertion has failed. If intInsert's value is 0, it has.
btnSave	Button	This is referenced by reference to the CheckAdditions() procedure: this is the btnSave from whichever form calls the procedure, i.e. the save button.
ict	String	The customer title in the WordInvoiceAutomationMagic() procedure, referenced by value from the invoice form, and this procedure uses its values in a Microsoft Word document.
icn	String	As above, but the customer name.
ica	String	As above, but the customer address.
icp	String	As above, but the customer postcode.
oWord	Word.Application	The Microsoft Word application object, which helps integrate Word into VB and enables the insertion of data into Microsoft Word documents.
oDoc	Word.Document	The Microsoft Word document object, enabling documents to be created from inside VB.
CustAddressStuff	Word.Paragraph	The customer address stuff, defined as a new paragraph in the Word document, and given values of all the variables passed by reference earlier for the values.
CustInvoiceHeader	Word.Paragraph	This variable's .Text element is just the header that is centred on the Word document's page that states that the document is an invoice.
CustQuotationHeader	Word.Paragraph	This variable's .Text element is just the header that is centred on the Word document's page that states that the document is a quotation.

Variable (cont.)	Data Type (cont.)	Purpose (cont.)
qct	String	The customer title in the WordQuotationAutomationMagic() procedure, referenced by value from the quotation form, and this procedure uses its values in a Microsoft Word document.
qcn	String	As above, but the customer name.
qca	String	As above, but the customer address.
qcp	String	As above, but the customer postcode.

(The final four variables in this table are the only new variables in the WordQuotationAutomationMagic() procedure—every other variable in that procedure also existed in WordInvoiceAutomationMagic() and does the same thing.)

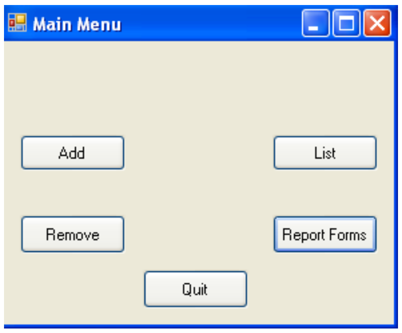
4.5 Annotated list of program code

4.5.1 frmLoginForm



```
ï»¿Imports System.Data
Imports System.Data.OleDb

Public Class frmLoginForm
    Public accConnection As New OleDbConnection
    'Connect to the Microsoft Access database.
    Dim ConnectionString As String = "Provider=Microsoft.Jet.OLEDB.4.0;" & _
            "Data Source = E:\SFS.mdb"

    Private Sub frmLoginForm_Load(ByVal sender As System.Object, _
                                  ByVal e As System.EventArgs) Handles MyBase.Load
        'Connection variable.
        accConnection = New OleDbConnection(ConnectionString)
    End Sub

    Private Sub OK_Click(ByVal sender As System.Object, _
                         ByVal e As System.EventArgs) Handles OK.Click

        'Get the login username and password from the database.
        Dim cmdString As String = "SELECT login_uname, login_pword FROM Login"
        Dim da As New OleDbDataAdapter(cmdString, accConnection)
        Dim ds As New DataSet
        Dim u As String = txtUsername.Text
        Dim p As String = txtPassword.Text

        da.Fill(ds, "Login")
        Dim dt As DataTable = ds.Tables(0)

        'The usernames and passwords obtained from the database, to
        'compare with the ones the user entered.
        Dim loginuname As String = ""
        Dim loginpword As String = ""

        If ds.Tables(0).Rows.Count <> 0 Then
            loginuname = ds.Tables(0).Rows(0).Item("login_uname")
            loginpword = ds.Tables(0).Rows(0).Item("login_pword")
        End If

        If u = loginuname And p = loginpword Then
            frmMainMenu.Show()
        Else
            MsgBox("Incorrect username or password.  Try again.")
            'Error, then blank the textboxes so that the user doesn't
            'have to.
            txtUsername.Text = ""
            txtPassword.Text = ""
        End If

    End Sub

    Private Sub Cancel_Click(ByVal sender As System.Object, _
                             ByVal e As System.EventArgs) Handles Cancel.Click
        Me.Close()
    End Sub

End Class
```

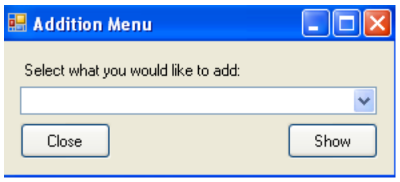
4.5.2 frmMainMenu



```
ï»¿Imports System.Data
Imports System.Data.OleDb
```

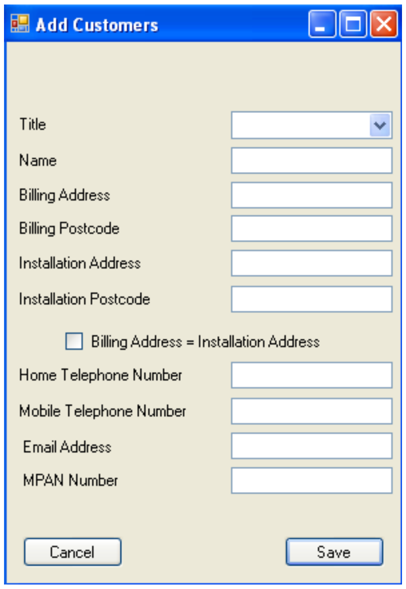
```
Public Class frmMainMenu

    'THIS CODE SATISFIES SPECIFIC OBJECTIVE 1.

    Private Sub frmMainMenu_Load(ByVal sender As System.Object, _
                                ByVal e As System.EventArgs) Handles MyBase.Load
        'Hide the login form from the user when this form is loaded so
        'that he/she is not having to close a load of windows when he/she
        'quits the program.
        frmLoginForm.Hide()
    End Sub

    'In each of these, according to the buttons clicked, show the correct
    'form and close this window.
    Private Sub btnReport_Click(ByVal sender As System.Object, _
                               ByVal e As System.EventArgs) Handles btnReport.Click
        Me.Hide()
        frmReportMenu.Show()
    End Sub

    Private Sub btnList_Click(ByVal sender As System.Object, _
                             ByVal e As System.EventArgs) Handles btnList.Click
        Me.Hide()
        frmList.Show()
    End Sub

    Private Sub btnAdd_Click(ByVal sender As System.Object, _
                            ByVal e As System.EventArgs) Handles btnAdd.Click
        Me.Hide()
        frmAdd.Show()
    End Sub

    Private Sub btnRemove_Click(ByVal sender As System.Object, _
                               ByVal e As System.EventArgs) Handles btnRemove.Click
        Me.Hide()
        frmRemove.Show()
    End Sub

    Private Sub btnQuit_Click(ByVal sender As System.Object, _
                             ByVal e As System.EventArgs) Handles btnQuit.Click
        Me.Close()
        'Now close the login form that is at present hidden so that the
        'program closes and doesn't stay running displaying nothing.  Also,
        'someone would have already logged in, so displaying the login
        'form again would intice people into logging in again, potentially
        'messing database stuff up like the closing of connections etc.
        frmLoginForm.Close()
    End Sub
End Class
```

4.5.3 frmAdd



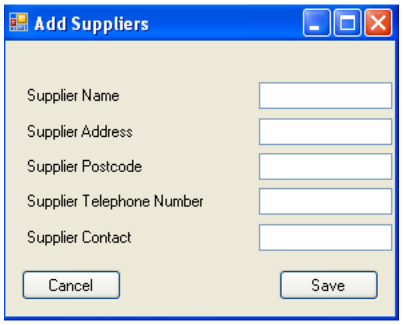
```
ï»¿Public Class frmAdd

    Private Sub frmAdd_Load(ByVal sender As System.Object, _
                            ByVal e As System.EventArgs) Handles MyBase.Load
        'Check if the database connection is breathing.
        'If it isn't, resuscitate it.  :)
        If frmLoginForm.accConnection.State <> ConnectionState.Open Then
            frmLoginForm.accConnection.Open()
        End If

        'Populate drop down box.
        cbtxtAddOptions.Items.Add("Customer")
        cbtxtAddOptions.Items.Add("Supplier")
        cbtxtAddOptions.Items.Add("Component")
    End Sub

    Private Sub btnShow_Click(ByVal sender As System.Object, _
                              ByVal e As System.EventArgs) Handles btnShow.Click

        'According to the option selected, display the next form and hide
        'this one.
        If cbtxtAddOptions.Text = "Customer" Then
            Me.Hide()
            frmAddCustomer.Show()
        ElseIf cbtxtAddOptions.Text = "Supplier" Then
            Me.Hide()
            frmAddSupplier.Show()
        ElseIf cbtxtAddOptions.Text = "Component" Then
            Me.Hide()
            frmAddComponent.Show()
        End If
    End Sub

    Private Sub btnClose_Click(ByVal sender As System.Object, _
                               ByVal e As System.EventArgs) Handles btnClose.Click
        'Close this window and show the main menu.
        Me.Close()
        frmMainMenu.Show()
    End Sub
End Class
```

4.5.4 frmAddCustomer



```
ï»¿Imports System.Data
Imports System.Data.OleDb

Public Class frmAddCustomer
    'THIS CODE SATISFIES SPECIFIC OBJECTIVE 3.
    Public accConnection As New OleDbConnection

    Private Sub frmAddCustomers_Load(ByVal sender As System.Object, _
                                     ByVal e As System.EventArgs) Handles MyBase.Load
        'Check if the database connection from the login form is still
        'alive - if it isn't, resuscitate (reopen) it.
        If frmLoginForm.accConnection.State <> ConnectionState.Open Then
            frmLoginForm.accConnection.Open()
        End If

        'Populate this list with selected titles.
        Me.cbCustomerTitle.Items.Add("Mr")
        Me.cbCustomerTitle.Items.Add("Mrs")
        Me.cbCustomerTitle.Items.Add("Miss")
        Me.cbCustomerTitle.Items.Add("Dr")

        'Add tooltip to lblCustomerName and txtCustomerName field
        'so that the user knows to add the name in a uniform way.
        Me.ttCustomerName.SetToolTip(Me.lblCustomerName, _
                                     "Name must be in the format 'Forename <space> Surname'.")
        Me.ttCustomerName.SetToolTip(Me.txtCustomerName, _
                                      "Name must be in the format 'Forename <space> Surname'.")
    End Sub

    Private Sub btnSave_Click(ByVal sender As System.Object, _
                              ByVal e As System.EventArgs) Handles btnSave.Click

        Dim cmdString As String = "INSERT INTO Customer (cust_title, cust_name, " _
                                  & "cust_billaddress, cust_billpostcode, cust_instaddress, " _
                                  & "cust_instpostcode, cust_hometelno, cust_mobtelno, " _
                                  & "cust_email, cust_mpan) VALUES (@cust_title,@cust_name," _
                                  & "@cust_billaddress,@cust_billpostcode,@cust_instaddress," _
                                  & "@cust_instpostcode,@cust_hometelno,@cust_mobtelno," _
                                  & "@cust_email,@cust_mpan)"

        Dim AddCustomerDataAdapter As New OleDbDataAdapter
        Dim accCommand As New OleDbCommand
        Dim intInsert As Integer
        Dim CustomerForename As String

        CustomerForename = txtCustomerName.Text
        accCommand.Connection = frmLoginForm.accConnection
        accCommand.CommandType = CommandType.Text
        accCommand.CommandText = cmdString

        'Make the customer bill address be the install address too if
        'the user ticks the box, and deactivate the installation address
        'textboxes.
        If cbCustomerBillInstallSameAddress.CheckState = 1 Then
            txtCustomerInstAddress.Text = txtCustomerBillAddress.Text
            txtCustomerInstPostcode.Text = txtCustomerBillPostcode.Text
            txtCustomerInstAddress.Enabled = False
            txtCustomerInstPostcode.Enabled = False
        End If

        'Call the insertion checking and inserting procedure.
        Call InsertParameters(accCommand)

        'Variable to check for errors - True if one exists, False if not.
        Dim error_yes_no As Boolean

        'Check if the email textbox contains '@', therefore if it is a
        'valid email address.
        Dim pos As Integer
        Dim etb As String = txtCustomerEmail.Text 'email textbox
        Dim atsymbol As String = "@"
        pos = InStr(etb, atsymbol)
        If pos = 0 Then
            error_yes_no = True
            intInsert = 0
            MsgBox("Input a valid email address!")
        Else
            error_yes_no = False
        End If

        'Check if there are any errors.  If there aren't, send it through
        'to the database.  Check for blank textboxes.
        Try
            If error_yes_no = False Then
                intInsert = accCommand.ExecuteNonQuery()
                btnSave.Enabled = False
            Else
                intInsert = 0
                MsgBox("There has been an error.")
            End If
        Catch ex As Exception
            intInsert = 0
            MsgBox("There has been an error.")
        End Try
    End Sub

    Private Sub InsertParameters(ByRef acccmd As OleDbCommand)
        'Insert the data into the database.

        acccmd.Parameters.Add("@cust_title", OleDbType.Char).Value = _
                                cbCustomerTitle.SelectedItem
        acccmd.Parameters.Add("@cust_name", OleDbType.Char).Value = _
                                txtCustomerName.Text
        acccmd.Parameters.Add("@cust_billaddress", OleDbType.Char).Value = _
                                txtCustomerBillAddress.Text
        acccmd.Parameters.Add("@cust_billpostcode", OleDbType.Char).Value = _
                                txtCustomerBillPostcode.Text
        acccmd.Parameters.Add("@cust_instaddress", OleDbType.Char).Value = _
                                txtCustomerInstAddress.Text
        acccmd.Parameters.Add("@cust_instpostcode", OleDbType.Char).Value = _
                                txtCustomerInstPostcode.Text
        acccmd.Parameters.Add("@cust_hometelno", OleDbType.Char).Value = _
                                txtCustomerHomeTelNo.Text
        acccmd.Parameters.Add("@cust_mobtelno", OleDbType.Char).Value = _
                                mtxtCustomerMobTelNo.Text
        acccmd.Parameters.Add("@cust_email", OleDbType.Char).Value = _
                                txtCustomerEmail.Text
        acccmd.Parameters.Add("@cust_mpan", OleDbType.Char).Value = _
                                txtCustomerMpanNo.Text
    End Sub

    Private Sub btnCancel_Click(ByVal sender As System.Object, _
                                ByVal e As System.EventArgs) Handles btnCancel.Click

        If btnSave.Enabled = False Then
            'It's all fine and has gone through OK, so don't display a
            'message because that would just be annoying, just close this
            'and display the previous form.
            Me.Close()
            frmAdd.Show()
        Else
            'If it hasn't gone through OK, or nothing has been added,
            'then let the user know this so as not to panic them;
            'if the user did not mean to click the button, he/she knows.
            MsgBox("This window will close and these details will not be saved.")
            Me.Close()
            frmAdd.Show()
        End If

    End Sub
End Class
```

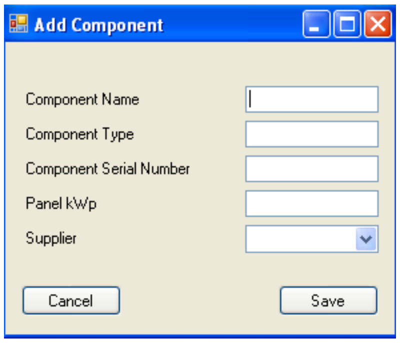
4.5.5 frmAddSupplier



```
ï»¿Imports System.Data
Imports System.Data.OleDb

Public Class frmAddSupplier
    'THIS CODE SATISFIES SPECIFIC OBJECTIVE 4.
    Public accConnection As New OleDbConnection

    Private Sub frmAddSupplier_Load(ByVal sender As System.Object, _
                                    ByVal e As System.EventArgs) Handles MyBase.Load
        If frmLoginForm.accConnection.State <> ConnectionState.Open Then
            frmLoginForm.accConnection.Open()
        End If
    End Sub

    Private Sub btnSave_Click(ByVal sender As System.Object, _
                              ByVal e As System.EventArgs) Handles btnSave.Click

        Dim cmdString As String = "INSERT INTO Supplier (supp_name, supp_address, supp_postcode, " _
                                & "supp_telno, supp_contactname)" _
                                & "VALUES (@supp_name,@supp_address,@supp_postcode," _
                                & "@supp_telno,@supp_contactname)"
        Dim AddSupplierDataAdapter As New OleDbDataAdapter
        Dim accCommand As New OleDbCommand
        Dim intInsert As Integer

        accCommand.Connection = frmLoginForm.accConnection
        accCommand.CommandType = CommandType.Text
        accCommand.CommandText = cmdString
        Call InsertParameters(accCommand)

        'Now check if all the textboxes are populated.
        Try
            intInsert = accCommand.ExecuteNonQuery()
        Catch ex As Exception
            intInsert = 0
            MsgBox("Enter a value in each of the boxes, and make it a valid one!")
        End Try

        'Call frmSwankyCode.CheckAdditions(intInsert, btnSave)

    End Sub

    Private Sub InsertParameters(ByRef acccmd As OleDbCommand)
        acccmd.Parameters.Add("@supp_name", OleDbType.Char).Value = txtSupplierName.Text
        acccmd.Parameters.Add("@supp_address", OleDbType.Char).Value = txtSupplierAddress.Text
        acccmd.Parameters.Add("@supp_postcode", OleDbType.Char).Value = txtSupplierPostCode.Text
        acccmd.Parameters.Add("@supp_telno", OleDbType.Char).Value = mtxtSupplierTelNo.Text
        acccmd.Parameters.Add("@supp_contactname", OleDbType.Char).Value = txtSupplierContactName.Text

    End Sub

    Private Sub btnCancel_Click(ByVal sender As System.Object, _
                                ByVal e As System.EventArgs) Handles btnCancel.Click

        If btnSave.Enabled = False Then
            'It's all fine and has gone through OK, so don't display a
            'message because that would just be annoying, just close this
            'and display the previous form.
            Me.Close()
            frmAdd.Show()
        Else
            'If it hasn't gone through OK, or nothing has been added,
            'then let the user know this so as not to panic them;
            'if the user did not mean to click the button, he/she knows.
            MsgBox("This window will close and these details will not be saved.")
            Me.Close()
            frmAdd.Show()
        End If

    End Sub
```

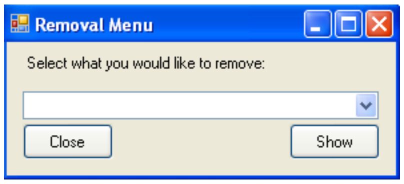
```
End Class
```

4.5.6 frmAddComponent



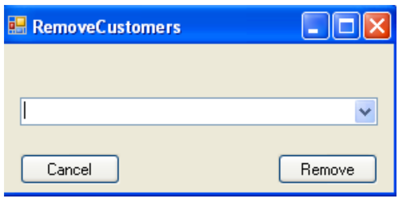
```
ï»¿Imports System.Data
Imports System.Data.OleDb

Class frmAddComponent
    'THIS CODE SATISFIES SPECIFIC OBJECTIVE 5.

    Public accConnection As New OleDbConnection

    Private Sub AddComponent_Load(ByVal sender As System.Object, _
                                  ByVal e As System.EventArgs) Handles MyBase.Load
        If frmLoginForm.accConnection.State <> ConnectionState.Open Then
            frmLoginForm.accConnection.Open()
        End If

        accConnection = frmLoginForm.accConnection
        Dim strSQL As String = "SELECT supp_name FROM Supplier"
        Dim da As New OleDbDataAdapter(strSQL, accConnection)
        Dim ds As New DataSet

        da.Fill(ds, "Supplier")

        Dim dt As DataTable = ds.Tables(0)
        Dim dr As DataRow

        For Each dr In dt.Rows()
            'List supplier names in the box so that the user can select
            'the supplier that the component is coming from.  Pull this
            'from the Supplier table.
            txtcbCompSupplierName.Items.Add(dr("supp_name"))
        Next

        txtcbCompSupplierName.SelectedIndex = -1
    End Sub

    Private Sub btnSave_Click(ByVal sender As System.Object, _
                              ByVal e As System.EventArgs) Handles btnSave.Click

        'This takes the supplier name from the supplier table and gets the
        'supplier ID from it, thereby letting it into the Component table.

        Dim suppidvaluecmd As String = "SELECT supp_id FROM Supplier WHERE " _
                                     & "supp_name = '" & txtcbCompSupplierName.SelectedItem & "'"

        Dim cmdString As String = "INSERT INTO Component (comp_name, comp_type, comp_serialno," _
                                & " comp_panelwp, comp_supplier)" _
                                & "VALUES (@comp_name,@comp_type,@comp_serialno," _
                                & "@comp_panelwp,@comp_supplier)"

        Dim da As New OleDbDataAdapter(suppidvaluecmd, accConnection)
        Dim ds As New DataSet
        da.Fill(ds, "Supplier")

        Dim AddSupplierDataAdapter As New OleDbDataAdapter
        Dim accCommand As New OleDbCommand
        Dim intInsert As Integer

        accCommand.Connection = frmLoginForm.accConnection
        accCommand.CommandType = CommandType.Text
        accCommand.CommandText = cmdString
        Dim supplieridvalue As Integer = ds.Tables(0).Rows(0).Item("supp_id")
        MsgBox(supplieridvalue)

        Call InsertParameters(accCommand, supplieridvalue)
        'Now check if all the textboxes are populated.
        Try
            intInsert = accCommand.ExecuteNonQuery()
        Catch ex As Exception
            MsgBox("Enter a value in each of the boxes, and make it a valid one!")
        End Try

        'Call frmSwankyCode.CheckAdditions(intInsert, btnSave)
    End Sub

    Private Sub InsertParameters(ByRef acccmd As OleDbCommand, ByRef siv As Integer)

        acccmd.Parameters.Add("@comp_name", OleDbType.Char).Value = txtComponentName.Text
        acccmd.Parameters.Add("@comp_type", OleDbType.Char).Value = txtComponentType.Text
        acccmd.Parameters.Add("@comp_serialno", OleDbType.Char).Value = txtComponentSerialNo.Text
        'Boundary data + erroneus data...
        Try
            If txtComponentPanelkWp.Text >= 0 Then
                'Multiply by 1000 to get the watt for the database rather than the
                'kilowatt value from the user.
                acccmd.Parameters.Add("@comp_panelwp", OleDbType.Numeric).Value = txtComponentPanelkWp.
                                                                        * 1000
            Else
                MsgBox("The panel's kWp cannot be negative.")
            End If

        Catch ex As Exception
            MsgBox("No numbers in the kWp box!")
        End Try
        'And now the supplier ID value.
        acccmd.Parameters.Add("@comp_supplier", OleDbType.Numeric).Value = siv
    End Sub

    Private Sub btnCancel_Click(ByVal sender As System.Object, _
                                ByVal e As System.EventArgs) Handles btnCancel.Click

        If btnSave.Enabled = False Then
            'It's all fine and has gone through OK, so don't display a
            'message because that would just be annoying, just close this
            'and display the previous form.
            Me.Close()
            frmAdd.Show()
        Else
            'If it hasn't gone through OK, or nothing has been added,
            'then let the user know this so as not to panic them;
            'if the user did not mean to click the button, he/she knows.
            MsgBox("This window will close and these details will not be saved.")
            Me.Close()
            frmAdd.Show()
        End If

    End Sub

End Class
```

4.5.7 frmRemove



```
ï»¿Public Class frmRemove

    Private Sub frmRemove_Load(ByVal sender As System.Object, _
                               ByVal e As System.EventArgs) Handles MyBase.Load
        'Check if the database connection is breathing.
        'If it isn't, resuscitate it.  :-)
        If frmLoginForm.accConnection.State <> ConnectionState.Open Then
            frmLoginForm.accConnection.Open()
        End If

        cbtxtRemoveOptions.Items.Add("Customer")
        cbtxtRemoveOptions.Items.Add("Supplier")
        cbtxtRemoveOptions.Items.Add("Component")
    End Sub

    Private Sub btnShow_Click(ByVal sender As System.Object, _
                              ByVal e As System.EventArgs) Handles btnShow.Click

        'According to the option selected, display the next form and hide
        'this one.
        If cbtxtRemoveOptions.Text = "Customer" Then
            Me.Hide()
            frmRemoveCustomer.Show()
        ElseIf cbtxtRemoveOptions.Text = "Supplier" Then
            Me.Hide()
            frmRemoveSupplier.Show()
        ElseIf cbtxtRemoveOptions.Text = "Component" Then
            Me.Hide()
            frmRemoveComponent.Show()
        End If

    End Sub

    Private Sub btnClose_Click(ByVal sender As System.Object, _
                               ByVal e As System.EventArgs) Handles btnClose.Click
        'Close this window and show the main menu.
        Me.Close()
        frmMainMenu.Show()
    End Sub
End Class
```

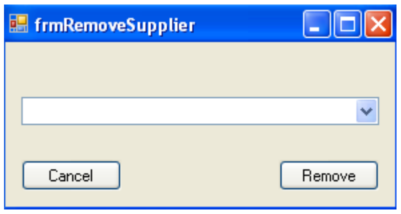
4.5.8 frmRemoveCustomer



```
ï»¿Imports System.Data
Imports System.Data.OleDb

Public Class frmRemoveCustomer
```

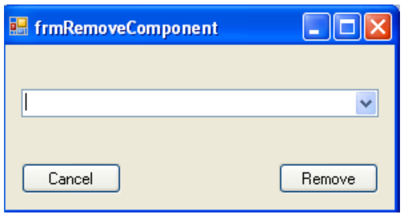
```
    'THIS CODE SATISFIES SPECIFIC OBJECTIVE 6.
    Public accConnection As New OleDbConnection

    Private Sub frmRemoveCustomers_Load(ByVal sender As System.Object, _
                                        ByVal e As System.EventArgs) Handles MyBase.Load
        If frmLoginForm.accConnection.State <> ConnectionState.Open Then
            frmLoginForm.accConnection.Open()
        End If

        accConnection = frmLoginForm.accConnection
        Dim strSQL As String = "SELECT cust_name FROM Customer"
        Dim da As New OleDbDataAdapter(strSQL, accConnection)
        Dim ds As New DataSet

        da.Fill(ds, "Customer")

        Dim dt As DataTable = ds.Tables(0)
        Dim dr As DataRow

        For Each dr In dt.Rows()
            txtcbCustomerRemove.Items.Add(dr("cust_name"))
        Next

        txtcbCustomerRemove.SelectedIndex = -1
    End Sub

    Private Sub btnRemove_Click(ByVal sender As System.Object, _
                                ByVal e As System.EventArgs) Handles btnRemove.Click

        Dim cmdString As String = "DELETE * FROM Customer WHERE cust_name = '" & _
                                  Me.txtcbCustomerRemove.SelectedItem & "'"
        Dim da As New OleDbDataAdapter(cmdString, accConnection)
        Dim ds As New DataSet
        Dim accCommand As New OleDbCommand

        accCommand.Connection = frmLoginForm.accConnection
        accCommand.CommandType = CommandType.Text
        accCommand.CommandText = cmdString

        Call RemoveParameters(accCommand)
        Dim intRemove As Integer

        intRemove = accCommand.ExecuteNonQuery()

        If intRemove = 0 Then
            MsgBox("Sorry, data deletion failed.")
            'Else, assume it went through OK.
        Else
            btnRemove.Enabled = False
        End If
    End Sub

    Private Sub RemoveParameters(ByRef acccmd As OleDbCommand)
        acccmd.Parameters.Add("@cust_name", OleDbType.Char).Value = _
                                Me.txtcbCustomerRemove.SelectedItem
    End Sub

    Private Sub btnCancel_Click(ByVal sender As System.Object, _
                                ByVal e As System.EventArgs) Handles btnCancel.Click
        Me.Close()
        frmRemove.Show()
    End Sub

End Class
```

4.5.9 frmRemoveSupplier



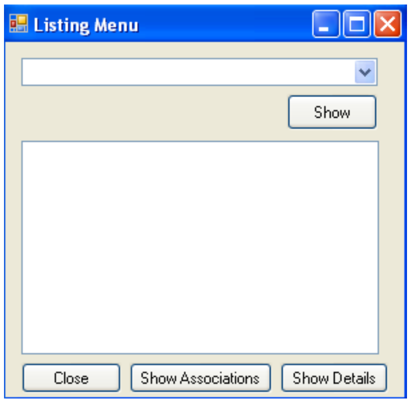
```
ï»¿Imports System.Data
Imports System.Data.OleDb

Public Class frmRemoveSupplier
    'THIS CODE SATISFIES SPECIFIC OBJECTIVE 7.
    Public accConnection As New OleDbConnection

    Private Sub frmRemoveSupplier_Load(ByVal sender As System.Object, _
                                       ByVal e As System.EventArgs) Handles MyBase.Load

        'Check if db connection is breathing.
        'If it isn't, resuscitate it.
        If frmLoginForm.accConnection.State <> ConnectionState.Open Then
            frmLoginForm.accConnection.Open()
        End If

        accConnection = frmLoginForm.accConnection
        Dim strSQL As String = "SELECT supp_id,supp_name FROM Supplier"
        Dim da As New OleDbDataAdapter(strSQL, accConnection)
        Dim ds As New DataSet

        da.Fill(ds, "Supplier")

        Dim dt As DataTable = ds.Tables(0)
        Dim dr As DataRow

        For Each dr In dt.Rows()
            'Populate the box with the supplier names.
            txtcbSupplierRemove.Items.Add(dr("supp_name"))
        Next

        txtcbSupplierRemove.SelectedIndex = -1

    End Sub

    Private Sub btnRemove_Click(ByVal sender As System.Object, _
                                ByVal e As System.EventArgs) Handles btnRemove.Click

        Dim RemoveSupplierDataAdapter As New OleDbDataAdapter
        Dim accCommand As New OleDbCommand
        Dim cmdString As String = "DELETE * FROM Supplier WHERE supp_name = " & _
                                    Me.txtcbSupplierRemove.SelectedItem & ""

        accCommand.Connection = frmLoginForm.accConnection
        accCommand.CommandType = CommandType.Text
        accCommand.CommandText = cmdString

        Call RemoveParameters(accCommand)
        'Execute the query to delete the record from the db.
        Dim intRemove As Integer

        Try
            intRemove = accCommand.ExecuteNonQuery()
        Catch ex As Exception
            intRemove = 0
            MsgBox("Maybe you should try deleting the associated component first?")
        End Try

        If intRemove = 0 Then
            MsgBox("Data deletion failed.")
            'Else, assume it went through OK.
        Else
            btnRemove.Enabled = False
        End If

    End Sub

    Private Sub RemoveParameters(ByRef acccmd As OleDbCommand)
        acccmd.Parameters.Add("@supp_name", OleDbType.Char).Value = _
                                    txtcbSupplierRemove.SelectedItem
    End Sub

    Private Sub btnCancel_Click(ByVal sender As System.Object, _
                                ByVal e As System.EventArgs) Handles btnCancel.Click
        Me.Close()
        frmRemove.Show()
    End Sub
End Class
```

4.5.10 frmRemoveComponent



```
ï»¿Imports System.Data
Imports System.Data.OleDb

Class frmRemoveComponent
    'THIS CODE SATISFIES SPECIFIC OBJECTIVE 8.
    Public accConnection As New OleDbConnection

    Private Sub frmRemoveComponent_Load(ByVal sender As System.Object, _
                                        ByVal e As System.EventArgs) Handles MyBase.Load
        'Check if the database connection is breathing.
        'If it isn't, resuscitate it.  :-)
        If frmLoginForm.accConnection.State <> ConnectionState.Open Then
            frmLoginForm.accConnection.Open()
        End If

        accConnection = frmLoginForm.accConnection
        Dim strSQL As String = "SELECT comp_name FROM Component"
        Dim da As New OleDbDataAdapter(strSQL, accConnection)
        Dim ds As New DataSet

        da.Fill(ds, "Component")

        Dim dt As DataTable = ds.Tables(0)
        Dim dr As DataRow

        For Each dr In dt.Rows()
            'List component names in the box.
            txtcbComponentRemove.Items.Add(dr("comp_name"))
        Next

        txtcbComponentRemove.SelectedIndex = -1

    End Sub

    Private Sub btnRemove_Click(ByVal sender As System.Object, _
                               ByVal e As System.EventArgs) Handles btnRemove.Click

        Dim cmdString As String = "DELETE * FROM Component WHERE comp_name = '" & _
                                   Me.txtcbComponentRemove.SelectedItem & "'"
        Dim da As New OleDbDataAdapter(cmdString, accConnection)
        Dim ds As New DataSet
        Dim accCommand As New OleDbCommand
        Dim intRemove As Integer

        accCommand.Connection = frmLoginForm.accConnection
        accCommand.CommandType = CommandType.Text
        accCommand.CommandText = cmdString

        intRemove = accCommand.ExecuteNonQuery()

        If intRemove = 0 Then
            MsgBox("Data deletion failed.")
            'Else, assume it went through OK.
        Else
            btnRemove.Enabled = False
        End If

    End Sub

    Private Sub RemoveParameters(ByRef acccmd As OleDbCommand)
        acccmd.Parameters.Add("@comp_name", OleDbType.Char).Value = _
                               txtcbComponentRemove.SelectedItem
    End Sub

    Private Sub btnCancel_Click(ByVal sender As System.Object, _
                               ByVal e As System.EventArgs) Handles btnCancel.Click
        Me.Close()
        frmRemove.Show()
    End Sub

End Class
```

4.5.11 frmList

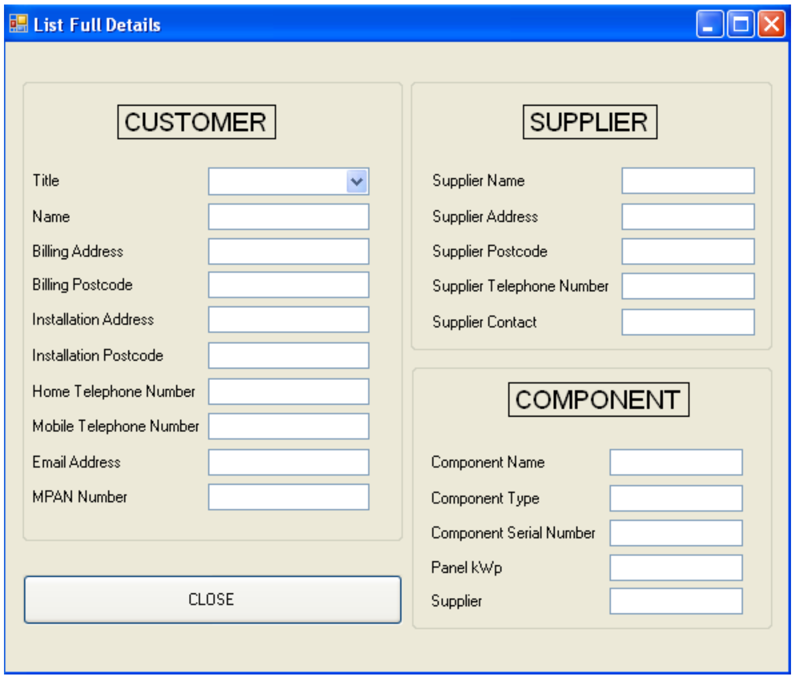


```
ï»¿Imports System.Data
Imports System.Data.OleDb

Public Class frmList
    'THIS CODE SATISFIES SPECIFIC OBJECTIVES 9, 10, 11.
    Public accConnection As New OleDbConnection

    'I think this is a better way of doing the listing [1] - fewer buttons
    'just a dropdown list which can be added to without having to create
    'another form

    '[1] - compared to the previous separate frmList[Customer|Supplier] etc.

    Private Sub frmList_Load(ByVal sender As System.Object, _
                             ByVal e As System.EventArgs) Handles MyBase.Load

        'Check if db connection is breathing.
        'If it isn't, resuscitate it.  :)
        If frmLoginForm.accConnection.State <> ConnectionState.Open Then
            frmLoginForm.accConnection.Open()
        End If

        'Populate the options list box with the following values.
        cbtxtListOptions.Items.Add("Customers")
        cbtxtListOptions.Items.Add("Suppliers")
        cbtxtListOptions.Items.Add("Components")
    End Sub

    Private Sub cbtxtListOptions_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.

        Dim cmdString As String = "" 'Initially blank.
        accConnection = frmLoginForm.accConnection
        Dim ds As New DataSet
        Dim dr As DataRow
        'da0, da1, da2
        'dt0, dt1, dt2
        'Declared within this if statement to make the program recognise
        'the change in cmdString and so know where to look.
        'Named da<num> and dt<num> to differentiate between different ifs.

        'Populate the list box with the result of the query - Customers.
        If cbtxtListOptions.Text = "Customers" Then
            cmdString = "SELECT cust_name FROM Customer"
            'Now clear the box if the selection changes.
            lbList.Items.Clear()
            Dim da0 As New OleDbDataAdapter(cmdString, accConnection)
            da0.Fill(ds, "Customer")
            Dim dt0 As DataTable = ds.Tables(0)
            For Each dr In dt0.Rows()
                lbList.Items.Add(dr("cust_name"))
            Next
            'Populate the list box with the result of the query - Suppliers.
        ElseIf cbtxtListOptions.Text = "Suppliers" Then
            cmdString = "SELECT supp_name FROM Supplier"
            'Now clear the box if the selection changes.
            lbList.Items.Clear()
            Dim da1 As New OleDbDataAdapter(cmdString, accConnection)
            da1.Fill(ds, "Supplier")
            Dim dt1 As DataTable = ds.Tables(0)
            For Each dr In dt1.Rows()
                lbList.Items.Add(dr("supp_name"))
            Next
            'Populate the list box with the result of the query - Components.
        ElseIf cbtxtListOptions.Text = "Components" Then
            cmdString = "SELECT comp_name FROM Component"
            'Now clear the box if the selection changes.
            lbList.Items.Clear()
            Dim da2 As New OleDbDataAdapter(cmdString, accConnection)
            da2.Fill(ds, "Component")
            Dim dt2 As DataTable = ds.Tables(0)
            For Each dr In dt2.Rows()
                lbList.Items.Add(dr("comp_name"))
            Next
        End If
    End Sub

    Private Sub btnSearch_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnS
        'Implement searching of customers, suppliers, and components.  OBJECTIVE 17.

        accConnection = frmLoginForm.accConnection
        Dim ds As New DataSet
        Dim dr As DataRow
        Dim strSQL As String = "" 'Initially empty.
        'The SQL query isn't searching for a straight string of the user
        'input, but variations of it, with 'LIKE'.
        If cbtxtListOptions.Text = "Customers" Then
            strSQL = "SELECT cust_name FROM Customer WHERE cust_name LIKE '%" & txtSearch.Text & "%'"
            'Clear the original Customer selection contents of lbList.
            lbList.Items.Clear()
            Dim da0 As New OleDbDataAdapter(strSQL, accConnection)
            da0.Fill(ds, "Customer")
            Dim dt0 As DataTable = ds.Tables(0)
            For Each dr In dt0.Rows()
                lbList.Items.Add(dr("cust_name"))
            Next
        ElseIf cbtxtListOptions.Text = "Suppliers" Then
            strSQL = "SELECT supp_name FROM Supplier WHERE supp_name LIKE '%" & txtSearch.Text & "%'"
            'Clear the original Supplier selection contents of lbList.
            lbList.Items.Clear()
            Dim da0 As New OleDbDataAdapter(strSQL, accConnection)
            da0.Fill(ds, "Supplier")
            Dim dt0 As DataTable = ds.Tables(0)
            For Each dr In dt0.Rows()
                lbList.Items.Add(dr("supp_name"))
            Next
        ElseIf cbtxtListOptions.Text = "Components" Then
            strSQL = "SELECT comp_name FROM Component WHERE comp_name LIKE '%" & txtSearch.Text & "%'"
            'Clear the original Component selection contents of lbList.
            lbList.Items.Clear()
            Dim da0 As New OleDbDataAdapter(strSQL, accConnection)
            da0.Fill(ds, "Components")
            Dim dt0 As DataTable = ds.Tables(0)
            For Each dr In dt0.Rows()
                lbList.Items.Add(dr("comp_name"))
            Next
        End If
    End Sub

    Private Sub btnShowAssoc_Click(ByVal sender As System.Object, _
                                   ByVal e As System.EventArgs) Handles btnShowAssoc.Click
        'THIS CODE SATISFIES SPECIFIC OBJECTIVE 13.
        'There are no associations between customers and suppliers,
        'or components and customers, at this stage.
        If cbtxtListOptions.SelectedItem = "Customers" Then
            MsgBox("No associations to show at this stage.")
        ElseIf cbtxtListOptions.SelectedItem = "Suppliers" Then
            MsgBox("No associations to show at this stage.")
        ElseIf cbtxtListOptions.SelectedItem = "Components" Then
            'Show the association between component and supplier: who supplies
            'each component.
            Call AssocCompSupp()
        End If
    End Sub
```

```
Private Sub AssocCompSupp()

    'THIS CODE SATISFIES SPECIFIC OBJECTIVE 13.

    'Variables here:
    'sn = supplier name;
    'cn = component name.

    accConnection = frmLoginForm.accConnection
    Dim cn As String = lbList.SelectedItem
    'Get the component's supplier ID from the Component table,
    'according to the component name.
    Dim compsuppliervalcmd As String = "SELECT comp_supplier FROM Component " & _
                                       "WHERE comp_name = '" & cn & "'"
    'Supplier name.
    Dim sn As String = ""

    Dim da As New OleDbDataAdapter(compsuppliervalcmd, accConnection)
    Dim ds As New DataSet
    da.Fill(ds, "Component")

    Dim compsupplierval As Integer = 0 'Initially.

    'Don't let the program crash if it detects that nothing has been
    'selected - just display the main form anyway.
    Try
        compsupplierval = ds.Tables(0).Rows(0).Item("comp_supplier")
    Catch
    End Try

    'Put the supplier ID value into another variable for ease of
    'use within the function just below.
    Dim csuppv As Integer = compsupplierval

    'Call the function to switch the supplier ID associated with the
    'component in the component table into the actual supplier name.
    'Send the suppname variable SN to the function so that it is
    'easier to reference here afterwards.
    Call SwitchSuppIDToName(csuppv, sn)

    'Make sure that totally blank "is supplied by" don't display
    'blank at both ends because the user hasn't selected an item.
    If lbList.SelectedIndex = -1 Then
        MsgBox("Cannot show associations; please select an item from the list.")
    Else
        MsgBox(cn & " is supplied by " & sn & ".")
    End If

End Sub
```

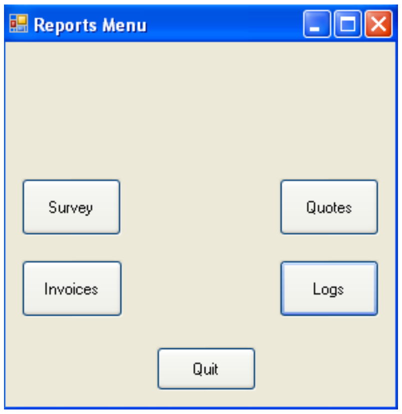
```
    Public Function SwitchSuppIDToName(ByVal csuppval As Integer, _
                                       ByRef finalsn As String)

        'This switches the obtained supplier ID to the supplier name.
        accConnection = frmLoginForm.accConnection
        Dim cmdString As String = "SELECT supp_name FROM Supplier WHERE supp_id = " & csuppval & ""

        Dim da As New OleDbDataAdapter(cmdString, accConnection)
        Dim ds As New DataSet
        da.Fill(ds, "Supplier")

        'Don't let the program crash if it detects that nothing has been
        'selected - just display the main form anyway.
        Try
            finalsn = ds.Tables(0).Rows(0).Item("supp_name")
        Catch
        End Try

        'Functions always return a value, so make this one return one.
        SwitchSuppIDToName = finalsn

    End Function

    Private Sub btnListFullDetails_Click(ByVal sender As System.Object, _
                                         ByVal e As System.EventArgs) Handles btnListFullDetails.Click
        frmListFullDetails.Show()
        Me.Close()
    End Sub

    Private Sub btnClose_Click(ByVal sender As System.Object, _
                               ByVal e As System.EventArgs) Handles btnClose.Click
        'Close this window and show the main menu.
        Me.Close()
        frmMainMenu.Show()
    End Sub

End Class
```

4.5.12 frmListFullDetails



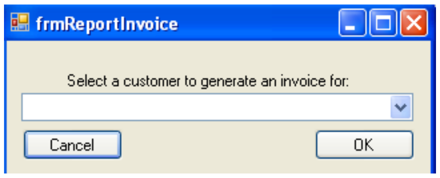
```
ï»¿Imports System.Data
Imports System.Data.OleDb

Public Class frmListFullDetails
    Public accConnection As New OleDbConnection

    Private Sub frmListFullDetails_Load(ByVal sender As System.Object, _
                                    ByVal e As System.EventArgs) Handles MyBase.Load
        'Reopen the database connection if it isn't open already.
        If frmLoginForm.accConnection.State <> ConnectionState.Open Then
            frmLoginForm.accConnection.Open()
        End If

        'Call different procedures depending on what the user selected.
        If frmList.cbtxtListOptions.SelectedItem = "Customers" Then
            Call PopulateCustInfo()
        ElseIf frmList.cbtxtListOptions.SelectedItem = "Suppliers" Then
            Call PopulateSuppInfo()
        ElseIf frmList.cbtxtListOptions.SelectedItem = "Components" Then
            Call PopulateCompInfo()
        End If
    End Sub

    Private Sub PopulateCustInfo()

        accConnection = frmLoginForm.accConnection
        'Input customer name from the other form and attribute it to CTN.
        Dim ctn As String = frmList.lbList.SelectedItem 'Customer name.
        Dim cmdString As String = "SELECT * FROM Customer WHERE cust_name = '" & ctn & "'"
        Dim accCommand As New OleDbCommand
        Dim da As New OleDbDataAdapter(cmdString, accConnection)
        Dim ds As New DataSet

        da.Fill(ds, "Customer")

        Dim dt As DataTable = ds.Tables(0)
        'Dim dr As DataRow <- This is redundant - it does not do anything.

        'Populate the textboxes in the form with existing customer details.
        'All the customer data text boxes are read-only to avoid data error:
        'to edit those, edit the customer details directly, then come back.

        'Populate the customer name textbox with the customer name
        'from the other form, before getting info from db about others.

        'An example of my variable naming here:
        'The variable 'txtldcn' = textbox listdetails custname.
        txtldctn.Text = ctn
        If ds.Tables(0).Rows.Count <> 0 Then
            cbldctt.Text = ds.Tables(0).Rows(0).Item("cust_title")
            txtldctba.Text = ds.Tables(0).Rows(0).Item("cust_billaddress")
            txtldctbp.Text = ds.Tables(0).Rows(0).Item("cust_billpostcode")
            txtldctia.Text = ds.Tables(0).Rows(0).Item("cust_instaddress")
            txtldctip.Text = ds.Tables(0).Rows(0).Item("cust_instpostcode")
            txtldcthtn.Text = ds.Tables(0).Rows(0).Item("cust_hometelno")
            txtldctmtn.Text = ds.Tables(0).Rows(0).Item("cust_mobtelno")
            txtldctea.Text = ds.Tables(0).Rows(0).Item("cust_email")
            txtldctmpan.Text = ds.Tables(0).Rows(0).Item("cust_mpan")
        End If

        accCommand.Connection = frmLoginForm.accConnection
        accCommand.CommandType = CommandType.Text
        accCommand.CommandText = cmdString
    End Sub

    Private Sub PopulateSuppInfo()

        accConnection = frmLoginForm.accConnection
        'Input supplier name from the other form and attribute it to SPN.
        Dim spn As String = frmList.lbList.SelectedItem 'Supplier name.
        Dim cmdString As String = "SELECT * FROM Supplier WHERE supp_name = '" & spn & "'"
        Dim accCommand As New OleDbCommand
        Dim da As New OleDbDataAdapter(cmdString, accConnection)
        Dim ds As New DataSet

        da.Fill(ds, "Supplier")
        Dim dt As DataTable = ds.Tables(0)
        'Dim dr As DataRow <- This is redundant - it does not do anything.

        'Populate the textboxes in the form with existing supplier details.
        'All the supplier data text boxes are read-only to avoid data error:
        'to edit those, edit the supplier details directly, then come back.

        'Populate the supplier name textbox with the customer name
        'from the other form, before getting info from db about others.

        'An example of my variable naming here:
        'The variable 'txtldspn' = textbox listdetails suppname.
        txtldspn.Text = spn
        If ds.Tables(0).Rows.Count <> 0 Then
            txtldspa.Text = ds.Tables(0).Rows(0).Item("supp_address")
            txtldspp.Text = ds.Tables(0).Rows(0).Item("supp_postcode")
            txtldsptn.Text = ds.Tables(0).Rows(0).Item("supp_telno")
            txtldspc.Text = ds.Tables(0).Rows(0).Item("supp_contactname")
        End If

        accCommand.Connection = frmLoginForm.accConnection
        accCommand.CommandType = CommandType.Text
        accCommand.CommandText = cmdString

    End Sub

    Private Sub PopulateCompInfo()

        accConnection = frmLoginForm.accConnection
        'Input component name from the other form and attribute it to CPN.
        Dim cpn As String = frmList.lbList.SelectedItem 'Component name.
        Dim cmdString As String = "SELECT * FROM Component WHERE comp_name = '" & cpn & "'"

        Dim accCommand As New OleDbCommand
        Dim da As New OleDbDataAdapter(cmdString, accConnection)
        Dim ds As New DataSet

        da.Fill(ds, "Component")

        Dim dt As DataTable = ds.Tables(0)
        'Dim dr As DataRow <- This is redundant - it does not do anything.

        Dim compsupplierval As Integer = 0

        'Check if a supplier is selected.  Before this Try/Catch, the
        'program would crash due to no value at row 0 of the table.
        Try
            compsupplierval = ds.Tables(0).Rows(0).Item("comp_supplier")
        Catch
        End Try
        Dim csuppv As Integer = compsupplierval
        Dim sn As String = "" 'Supplier name, initially blank here.

        'Now call the List form's switching supplier ID function.
        'No point creating a near identical one if there exists the
        'ability of reusing code.
        Call frmList.SwitchSuppIDToName(csuppv, sn)

        'Populate the textboxes in the form with existing component details.
        'All the component data text boxes are read-only to avoid data error:
        'to edit those, edit the component details directly, then come back.

        'Populate the component name textbox with the customer name
        'from the other form, before getting info from db about others.

        'An example of my variable naming here:
        'The variable 'txtldcpn' = textbox listdetails compname.
        txtldcpn.Text = cpn
        If ds.Tables(0).Rows.Count <> 0 Then
            txtldcpt.Text = ds.Tables(0).Rows(0).Item("comp_type")
            txtldcpsn.Text = ds.Tables(0).Rows(0).Item("comp_serialno")
            'Divide by 1000 to get the kilowatt for the user rather than
            'the watt value from the database.
            txtldcppkwp.Text = ds.Tables(0).Rows(0).Item("comp_panelwp") / 1000
            txtldcpspn.Text = sn
        End If

        accCommand.Connection = frmLoginForm.accConnection
        accCommand.CommandType = CommandType.Text
        accCommand.CommandText = cmdString
    End Sub

    Private Sub btnClose_Click(ByVal sender As System.Object, _
                               ByVal e As System.EventArgs) Handles btnClose.Click
        frmList.Show()
        Me.Close()
    End Sub

End Class
```

4.5.13 frmReportMenu



```
ï»¿Public Class frmReportMenu

    Private Sub btnQuit_Click(ByVal sender As System.Object, _
                              ByVal e As System.EventArgs) Handles btnQuit.Click
        'On close, close this window and show the main menu.
        Me.Close()
        frmMainMenu.Show()
    End Sub

    Private Sub btnQuotes_Click(ByVal sender As System.Object, _
                                ByVal e As System.EventArgs) Handles btnQuotes.Click
        'Show the quote form to the user, then hide this menu window.
        frmReportQuote.Show()
        Me.Hide()
    End Sub

    Private Sub btnLogs_Click(ByVal sender As System.Object, _
                              ByVal e As System.EventArgs) Handles btnLogs.Click
        'Show the log form to the user, then hide this menu window.
        frmReportLog.Show()
        Me.Hide()
    End Sub

    Private Sub btnInvoices_Click(ByVal sender As System.Object, _
                                  ByVal e As System.EventArgs) Handles btnInvoices.Click
        'Show the invoice form to the user, then hide this menu window.
        frmReportInvoice.Show()
        Me.Hide()
    End Sub

    Private Sub btnSurvey_Click(ByVal sender As System.Object, _
                                ByVal e As System.EventArgs) Handles btnSurvey.Click
        'Show the survey form to the user, then hide this menu window.
        frmReportSurvey.Show()
        Me.Hide()
    End Sub

End Class
```

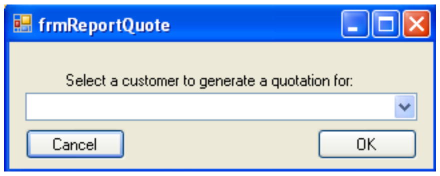
4.5.14 frmReportInvoice



Select a customer to generate an invoice for:

```
ï»¿Imports System.Data
Imports System.Data.OleDb

Class frmReportInvoice
    'THIS CODE SATISFIES SPECIFIC OBJECTIVE 12.
    Public accConnection As New OleDbConnection

    Private Sub frmReportInvoice_Load(ByVal sender As System.Object, _
                                      ByVal e As System.EventArgs) Handles MyBase.Load

        If frmLoginForm.accConnection.State <> ConnectionState.Open Then
            frmLoginForm.accConnection.Open()
        End If

        accConnection = frmLoginForm.accConnection
        Dim strSQL As String = "SELECT cust_name FROM Customer"
        Dim da As New OleDbDataAdapter(strSQL, accConnection)
        Dim ds As New DataSet

        da.Fill(ds, "Customer")

        Dim dt As DataTable = ds.Tables(0)
        Dim dr As DataRow

        For Each dr In dt.Rows()
            'List customer names in the box so that the user can select
            'a customer to show the invoice of.  At least customer details
            'will be inserted into the Word document at this stage.
            txtcbCustNameForInvoice.Items.Add(dr("cust_name"))
        Next

        txtcbCustNameForInvoice.SelectedIndex = -1

    End Sub


    Private Sub btnOK2_Click(ByVal sender As System.Object, _
                                  ByVal e As System.EventArgs) Handles btnOK2.Click

        accConnection = frmLoginForm.accConnection
        'Input supplier name from the other form and attribute it to CTN.
        Dim ctn As String = txtcbCustNameForInvoice.SelectedItem
        Dim cmdString As String = "SELECT * FROM Customer WHERE cust_name = '" & ctn & "'"

        Dim accCommand As New OleDbCommand
        Dim da As New OleDbDataAdapter(cmdString, accConnection)
        Dim ds As New DataSet

        'Validation.
        If txtcbCustNameForInvoice.SelectedItem = "" Then
            MsgBox("Select a customer to generate an invoice for!")
        Else
            da.Fill(ds, "Customer")

            Dim dt As DataTable = ds.Tables(0)

            'Now create variables to hold the useful address values of the
            ''* FROM Customer' so they'll be useful later on for the Word
            'invoicing stuff.
            'aicn, for example, = add invoice cust name
            Dim aict As String = ds.Tables(0).Rows(0).Item("cust_title")
            Dim aicn As String = ds.Tables(0).Rows(0).Item("cust_name")
            'Billing address and postcode in this case.
            Dim aica As String = ds.Tables(0).Rows(0).Item("cust_billaddress")
            Dim aicp As String = ds.Tables(0).Rows(0).Item("cust_billpostcode")

            'Now for some Word automation magic.  Call a procedure to do this
            'to save cluttering this OK button's execution with code.
            Call frmSwankyCode.WordInvoiceAutomationMagic(aict, aicn, aica, aicp)

        End If

    End Sub

    Private Sub btnCancel_Click(ByVal sender As System.Object, _
                                ByVal e As System.EventArgs) Handles btnCancel.Click
        frmReportMenu.Show()
        Me.Close()
    End Sub
End Class
```

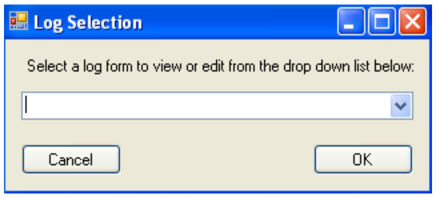
4.5.15 frmReportQuote



Select a customer to generate a quotation for:

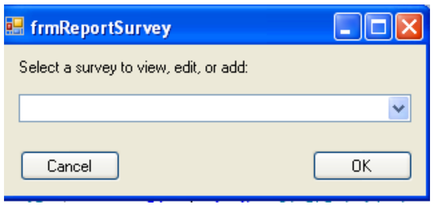
```
Imports System.Data
Imports System.Data.OleDb

Public Class frmReportQuote
    'THIS CODE SATISFIES SPECIFIC OBJECTIVE 12.
    Public accConnection As New OleDbConnection

    Private Sub frmReportQuote_Load(ByVal sender As System.Object, _
                                    ByVal e As System.EventArgs) Handles MyBase.Load

        If frmLoginForm.accConnection.State <> ConnectionState.Open Then
            frmLoginForm.accConnection.Open()
        End If

        accConnection = frmLoginForm.accConnection
        Dim strSQL As String = "SELECT cust_name FROM Customer"
        Dim da As New OleDbDataAdapter(strSQL, accConnection)
        Dim ds As New DataSet

        da.Fill(ds, "Customer")

        Dim dt As DataTable = ds.Tables(0)
        Dim dr As DataRow

        For Each dr In dt.Rows()
            'List customer names in the box so that the user can select
            'a customer to show the quotation of.  At least customer details
            'will be inserted into the Word document at this stage.
            txtcbCustNameForQuotation.Items.Add(dr("cust_name"))
        Next

        txtcbCustNameForQuotation.SelectedIndex = -1

    End Sub

    Private Sub btnCancel_Click(ByVal sender As System.Object, _
                                ByVal e As System.EventArgs) Handles btnCancel.Click
        Me.Close()
        frmReportMenu.Show()
    End Sub

    Private Sub btnOK2_Click(ByVal sender As System.Object, _
                                  ByVal e As System.EventArgs) Handles btnOK2.Click

        accConnection = frmLoginForm.accConnection
        'Input customer name from the other form and attribute it to CTN.
        Dim ctn As String = txtcbCustNameForQuotation.SelectedItem
        Dim cmdString As String = "SELECT * FROM Customer WHERE cust_name = '" & ctn & "'"

        Dim accCommand As New OleDbCommand
        Dim da As New OleDbDataAdapter(cmdString, accConnection)
        Dim ds As New DataSet

        da.Fill(ds, "Customer")

        Dim dt As DataTable = ds.Tables(0)

        'Now create variables to hold the useful address values of the
        ''* FROM Customer' so they'll be useful later on for the Word
        'quotation stuff.
        'aqcn, for example, = add quotation cust name
        Dim aqct As String = ds.Tables(0).Rows(0).Item("cust_title")
        Dim aqcn As String = ds.Tables(0).Rows(0).Item("cust_name")
        'Billing address and postcode in this case.
        Dim aqca As String = ds.Tables(0).Rows(0).Item("cust_billaddress")
        Dim aqcp As String = ds.Tables(0).Rows(0).Item("cust_billpostcode")

        'Now for some Word automation magic.  Call a procedure to do this
        'to save cluttering this OK button's execution with code.
        Call frmSwankyCode.WordQuotationAutomationMagic(aqct, aqcn, aqca, aqcp)

    End Sub
End Class
```

4.5.16 frmReportLog


Log Selection
Select a log form to view or edit from the drop down list below:
Cancel
OK

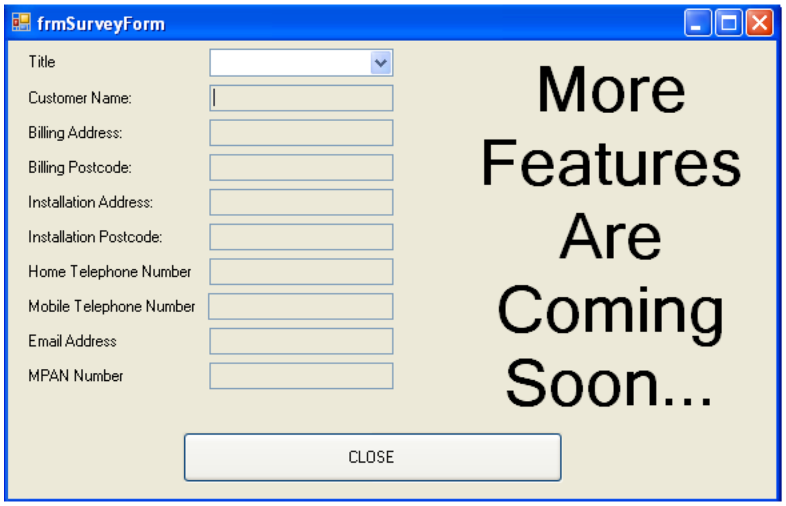
```
ï»¿Imports Microsoft.Office.Core
Imports Microsoft.Office.Interop
Imports Microsoft.Office.Interop.Word

Public Class frmReportLog
    'THIS CODE SATISFIES SPECIFIC OBJECTIVE 12.

    Private Sub frmReportLog_Load(ByVal sender As System.Object, _
                                        ByVal e As System.EventArgs) Handles MyBase.Load

        'Drop down menu populated with months up to 12/12.
        txtcbReportLog.Items.Add("December 2011")
        txtcbReportLog.Items.Add("January 2012")
        txtcbReportLog.Items.Add("February 2012")
        txtcbReportLog.Items.Add("March 2012")
        txtcbReportLog.Items.Add("April 2012")
        txtcbReportLog.Items.Add("May 2012")
        txtcbReportLog.Items.Add("June 2012")
        txtcbReportLog.Items.Add("July 2012")
        txtcbReportLog.Items.Add("August 2012")
        txtcbReportLog.Items.Add("September 2012")
        txtcbReportLog.Items.Add("October 2012")
        txtcbReportLog.Items.Add("November 2012")
        txtcbReportLog.Items.Add("December 2012")

    End Sub

    Private Sub btnOK_Click(ByVal sender As System.Object, _
                            ByVal e As System.EventArgs) Handles btnOK.Click
        'On click, open the selected file from the drop down list with
        'Microsoft Word.

        Dim ms_word As New Word.Application
        Dim worddoc As New Word.Document

        'Make the name shorter purely for ease of typing, despite Visual
        'Studio's tab completion.
        Dim rlsi As String = txtcbReportLog.SelectedItem

        'Now open the correct log form.
        'Get the left and right characters of the string 'rlsi'.
        'Make a new string out of those (in the format
        'E:\sfsstuff\log_[beginninglettersofmonth][enddigitsofyear].docx)
        'and put it directly into the doc opening, i.e. log_dec11.docx.
                'Make the filename all lowercase.
        Dim filename As String = Strings.Left(rlsi, 3) + Strings.Right(rlsi, 2)
        ms_word.Documents.Open("E:\sfsstuff\log_" + Strings.LCase(filename) + ".docx")

        'Let the user see the document when it opens.
        ms_word.WindowState = Word.WdWindowState.wdWindowStateNormal
        ms_word.Visible = True

    End Sub

    Private Sub btnCancel_Click(ByVal sender As System.Object, _
                                ByVal e As System.EventArgs) Handles btnCancel.Click
        Me.Close()
        frmReportMenu.Show()
    End Sub
End Class
```

4.5.17 frmReportSurvey



```
Imports System.Data
Imports System.Data.OleDb

Public Class frmReportSurvey

    Private Sub frmReportSurvey_Load(ByVal sender As System.Object, _
                                     ByVal e As System.EventArgs) Handles MyBase.Load
        Dim cmdString As String = "SELECT cust_name FROM Customer"
        'check if db connection is breathing
        'if it isn't, resuscitate it :)
        If frmLoginForm.accConnection.State <> ConnectionState.Open Then
            frmLoginForm.accConnection.Open()
        End If
        Dim accConnection As OleDbConnection = frmLoginForm.accConnection
        Dim da As New OleDbDataAdapter(cmdString, accConnection)
        Dim ds As New DataSet

        da.Fill(ds, "Customer")

        Dim dt As DataTable = ds.Tables(0)
        Dim dr As DataRow

        For Each dr In dt.Rows()
            txtcbReportSurvey.Items.Add(dr("cust_name"))
        Next

        txtcbReportSurvey.SelectedIndex = -1

    End Sub


    Private Sub btnOK_Click(ByVal sender As System.Object, _
                            ByVal e As System.EventArgs) Handles btnOK.Click

        Dim custname As String = txtcbReportSurvey.SelectedItem
        frmSurveyForm.Show()
    End Sub
    Private Sub btnCancel_Click(ByVal sender As System.Object, _
                                ByVal e As System.EventArgs) Handles btnCancel.Click
        Me.Close()
        frmReportMenu.Show()
    End Sub
End Class
```

4.5.18 frmSurveyForm



```
Imports System.Data
Imports System.Data.OleDb
Public Class frmSurveyForm

    Public accConnection As New OleDbConnection

    Private Sub frmSurveyForm_Load(ByVal sender As System.Object, _
                                   ByVal e As System.EventArgs) Handles MyBase.Load
        'All this does, for now, is display the selected customer details.
        'In the future, full survey details will be able to be entered by
        'the user, however at this moment I cannot implement this due to
        'time constraints, so it will be implemented in after-development.
        'To show this to the user when he or she hunts around, there
        'exists a placeholder text label that states that features are
        'coming soon.

        If frmLoginForm.accConnection.State <> ConnectionState.Open Then
            frmLoginForm.accConnection.Open()
        End If

        accConnection = frmLoginForm.accConnection
        'Input customer name from the other form and attribute it to CN.
        Dim cn As String = frmReportSurvey.txtcbReportSurvey.SelectedItem
        Dim cmdString As String = "SELECT * FROM Customer WHERE cust_name = '" & cn & "'"
        Dim accCommand As New OleDbCommand
        Dim da As New OleDbDataAdapter(cmdString, accConnection)
        Dim ds As New DataSet

        da.Fill(ds, "Customer")

        Dim dt As DataTable = ds.Tables(0)
        Dim dr As DataRow '<- This is redundant.sss

        'Populate the textboxes in the form with existing customer details.
        'All the customer data text boxes are read-only to avoid data error:
        'to edit those, edit the customer details directly, then come back.

        'Populate the customer name textbox with the customer name
        'from the other form, before getting info from db about others.
        txtscn.Text = cn
        If ds.Tables(0).Rows.Count <> 0 Then
            cbsct.Text = ds.Tables(0).Rows(0).Item("cust_title")
            txtscba.Text = ds.Tables(0).Rows(0).Item("cust_billaddress")
            txtscbp.Text = ds.Tables(0).Rows(0).Item("cust_billpostcode")
            txtscia.Text = ds.Tables(0).Rows(0).Item("cust_instaddress")
            txtscip.Text = ds.Tables(0).Rows(0).Item("cust_instpostcode")
            txtschtn.Text = ds.Tables(0).Rows(0).Item("cust_hometelno")
            txtscmtn.Text = ds.Tables(0).Rows(0).Item("cust_mobtelno")
            txtscmpan.Text = ds.Tables(0).Rows(0).Item("cust_mpan")
            txtscea.Text = ds.Tables(0).Rows(0).Item("cust_email")
        End If

        accCommand.Connection = frmLoginForm.accConnection
        accCommand.CommandType = CommandType.Text
        accCommand.CommandText = cmdString

    End Sub

    Private Sub btnClose_Click(ByVal sender As System.Object, _
                               ByVal e As System.EventArgs) Handles btnClose.Click
        Me.Close()
        frmReportSurvey.Show()
    End Sub
End Class
```

4.5.19 frmSwankyCode

I have not included a screenshot here as it would provide no information whatsoever due to the form being blank and serving no function other than a home for stray but useful pieces of code.

```
Imports Microsoft.Office.Core
Imports Microsoft.Office.Interop
Imports Microsoft.Office.Interop.Word

Public Class frmSwankyCode

    'This is a form because I couldn't work out how to make it work any
    'other way.

    'This is where all the code goes that is referenced from other subs
    'in other forms, to save space and attempt to minimise repetition.

    Public Sub CheckAdditions(ByVal intInsert As Integer, _
                              ByRef btnSave As Button)

        If intInsert = 0 Then
            MsgBox("Data insertion failed.")
        Else
            'Assume it went through OK and disable the Save button to
            'guard against duplication.
            btnSave.Enabled = False
        End If

    End Sub

    Public Sub WordInvoiceAutomationMagic(ByVal ict As String, ByVal icn As String, _
                                          ByVal ica As String, ByVal icp As String)

        'THIS CODE SATISFIES SPECIFIC OBJECTIVE 14.

        Dim oWord As Word.Application
        Dim oDoc As Word.Document
        Dim CustAddressStuff As Word.Paragraph
        Dim CustInvoiceHeader As Word.Paragraph

        'Start Word and open the document.
        'This is object stuff that I don't understand, but it was found on
        'a Microsoft doc, and the other stuff I tried didn't work, so I
        'used it.
        oWord = CreateObject("Word.Application")
        oWord.Visible = True
        oDoc = oWord.Documents.Add

        'Insert the customer billing address stuff.
        CustAddressStuff = oDoc.Content.Paragraphs.Add
        CustAddressStuff.Range.Text = ict & " " & icn & Chr(11) & ica _
                                      & Chr(11) & icp & Chr(11)
        CustAddressStuff.Range.ParagraphFormat.Alignment = _
            Word.WdParagraphAlignment.wdAlignParagraphRight
        CustAddressStuff.Range.Font.Bold = False

        'Sort out the spacing afterwards.
        CustAddressStuff.Format.SpaceAfter = 4
        CustAddressStuff.Range.InsertParagraphAfter()

        'Now, after the spacing, the INVOICE header, centred.
        CustInvoiceHeader = _
            oDoc.Content.Paragraphs.Add(oDoc.Bookmarks.Item("\endofdoc").Range)
        CustInvoiceHeader.Range.Text = "INVOICE"
        CustInvoiceHeader.Range.ParagraphFormat.Alignment = _
            Word.WdParagraphAlignment.wdAlignParagraphCenter
        CustInvoiceHeader.Range.Font.Size = 22
        CustInvoiceHeader.Range.Font.Bold = True

    End Sub

    Public Sub WordQuotationAutomationMagic(ByVal qct As String, ByVal qcn As String, _
                                ByVal qca As String, ByVal qcp As String)

        'THIS CODE SATISFIES SPECIFIC OBJECTIVE 14.

        Dim oWord As Word.Application
        Dim oDoc As Word.Document
        Dim CustAddressStuff As Word.Paragraph
        Dim CustQuotationHeader As Word.Paragraph

        'Start Word and open the document.
        'This is object stuff that I don't understand, but it was found on
        'a Microsoft doc, and the other stuff I tried didn't work, so I
        'used it.
        oWord = CreateObject("Word.Application")
        oWord.Visible = True
        oDoc = oWord.Documents.Add

        'Insert the customer billing address stuff.
        CustAddressStuff = oDoc.Content.Paragraphs.Add
        CustAddressStuff.Range.Text = qct & " " & qcn & Chr(11) & qca _
                                      & Chr(11) & qcp & Chr(11)
        CustAddressStuff.Range.ParagraphFormat.Alignment = _
            Word.WdParagraphAlignment.wdAlignParagraphRight
        CustAddressStuff.Range.Font.Bold = False

        'Sort out the spacing afterwards.
        CustAddressStuff.Format.SpaceAfter = 4
        CustAddressStuff.Range.InsertParagraphAfter()

        'Now, after the spacing, the Quotation header, centred.
        CustQuotationHeader = _
            oDoc.Content.Paragraphs.Add(oDoc.Bookmarks.Item("\endofdoc").Range)
        CustQuotationHeader.Range.Text = "QUOTATION"
        CustQuotationHeader.Range.ParagraphFormat.Alignment = _
            Word.WdParagraphAlignment.wdAlignParagraphCenter
        CustQuotationHeader.Range.Font.Size = 22
        CustQuotationHeader.Range.Font.Bold = True

    End Sub
End Class
```

Chapter 5

User Manual

Contents

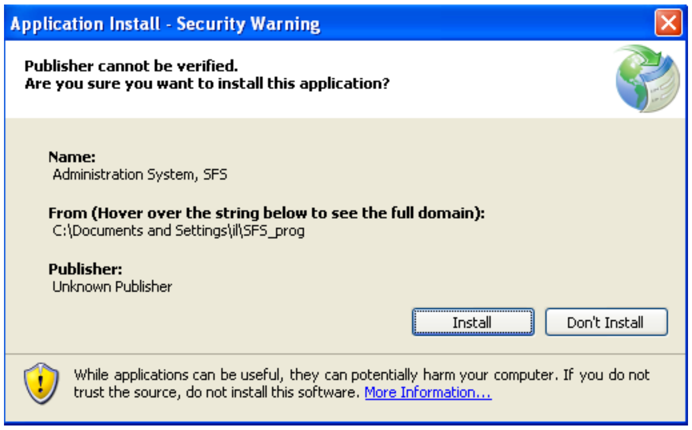
5.1 Introduction

This user manual will explain how to use the system that has been created for you, the user. If anything is not clear, please contact me, or attempt to work the functioning out! My contact details are provided in a separate, user-only document.

5.2 Usage instructions

5.2.1 Installing the program

The first step is the installation of the program! To do this, navigate to the folder view of the USB stick that the program will be provided on, and click setup. A window like this will display:



Application Install - Security Warning

Publisher cannot be verified.
Are you sure you want to install this application?

Name:
Administration System, SFS

From (Hover over the string below to see the full domain):
C:\Documents and Settings\il\SFS_prog

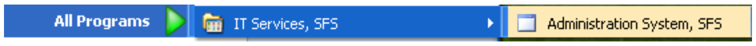
Publisher:
Unknown Publisher

While applications can be useful, they can potentially harm your computer. If you do not trust the source, do not install this software. More Information...

Click Install, ignoring any warnings about untrusted publishers. The program will then install reasonably quickly.

5.2.2 Launching the program

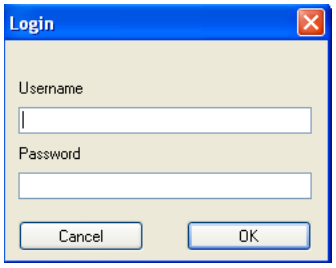
To use the program, click Start, then the below:



5.2.3 Logging in

When the program has successfully launched, the login form will display. This form demands a username and password: any of the ones provided in the table below may be used. Enter the username in the username textbox indicated, and the password into the password textbox, then click on or tab to the OK button.

Username	**Password**
sfs	sfs



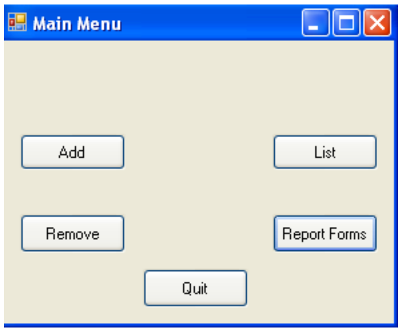
If the login attempt is unsuccessful, an error message box will display. Try entering the details again. If the error persists, contact me—it may be due to a typo in the database or this document.

If everything is fine, either on the first, second, or *n*th attempt, the login form should hide itself and the main menu should appear.

5.2.4 The main menu

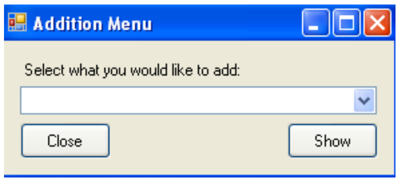
The main menu is the core from which everything can be accessed.

The menu, as below, contains buttons that can be clicked to access various parts of the program. This manual will explain each of them in turn.



5.2.5 Adding customers, suppliers, or components

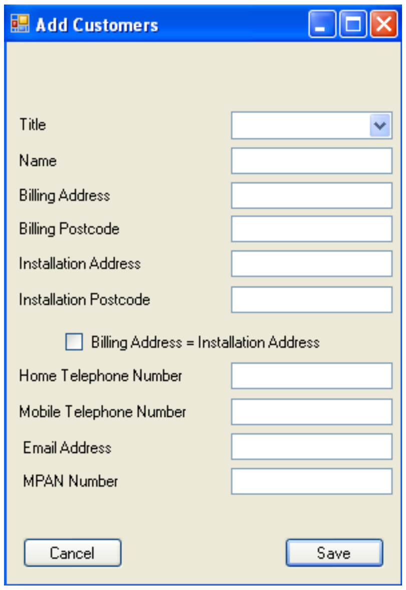
Clicking the add button from the main menu will open the Add form, where it is possibe to select what to add: either customer, supplier, or component, as below.



Click on the dropdown box to select either Customer, Supplier, or Component, then click OK. The appropriate form will display. Read on. . . .

5.2.5.1 Customers

If Customers is chosen, the add customer form appears:.



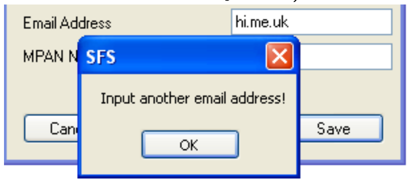
Input the details, then click Save. If there is an error in the details, the program will give an error message and data amendment is possible. If there is no error and therefore the data insertion is successful, the save textbox will turn grey, signifying that the form can be closed safely.

Clicking the Cancel button will close the form, thereby not saving any of the data entered into the textboxes.

Some things to note:

- If there is no data to input into one textbox at the time of addition, input 'NULL', do not just leave the textbox blank: doing so will display an error.
- The customer name must be in the format *Forename Surname*—there are not separate textboxes for forename and surname.
- UK telephone numbers must contain spaces between them, e.g. *01234 567890*, NOT *01234567890* or variants.
- If the customer's installation address is the same as their already entered billing address, tick the box: this will automatically populate the installation address fields in the form and the database.
- Email addresses must contain the '@' symbol.
- The MPAN number cannot exceed 14 characters.

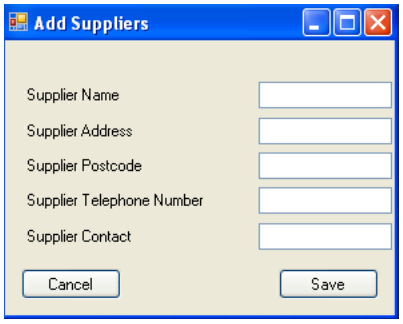
If the email address is seen as invalid (i.e. it does not contain an '@' symbol), the below error will display, and similar errors will appear for other erroneous data:



Input another email address!

5.2.5.2 Suppliers

If Suppliers is chosen, the add supplier form appears.



As above with the customer details, input the details, then click Save. If there is an error in the details, the program will give an error message and data amendment is possible. If there is no error and therefore the data insertion is successful, the save textbox will turn grey, signifying inactivity.

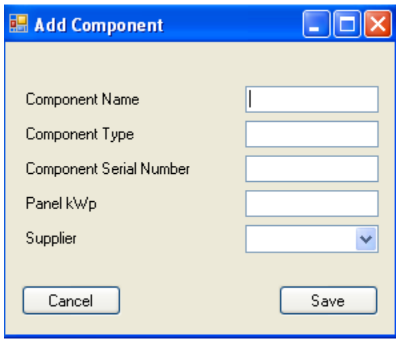
Clicking the Cancel button will close the form, thereby not saving any of the data entered into the textboxes.

Some things to note:

- If there is no data to input into one textbox at the time of addition, input 'NULL', do not just leave the textbox blank: doing so will display an error.
- UK telephone numbers must contain spaces between them, e.g. *01234 567890*, NOT *01234567890* or variants.
- The Supplier Contact textbox is the container for the name of the person who most frequently deals with communication with Sunny Future Solar at the supplier. For example: Ben at Alternergy.

5.2.5.3 Components

If Components is chosen, the add component form appears.



Input the details, then click Save. If there is an error in the details, the program will give an error message and data amendment is possible. If there is no error and therefore the data insertion is successful, the save textbox will turn grey, signifying inactivity.

Clicking the Cancel button will close the form, thereby not saving any of the data entered into the textboxes.

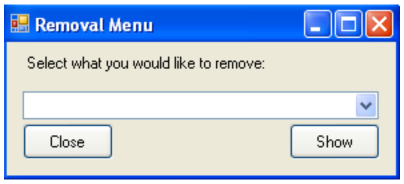
Some things to note:

- If there is no data to input into one textbox at the time of addition, input 'NULL', do not just leave the textbox blank: doing so will display an error.

- The textbox labelled ‘Supplier’ in this form should be used to specify which supplier supplies the added component. If the supplier for the component does not exist at the time, it must be added before adding the component itself, using the supplier addition form documented above.

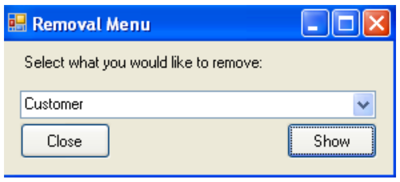
5.2.6 Removing customers, suppliers, or components

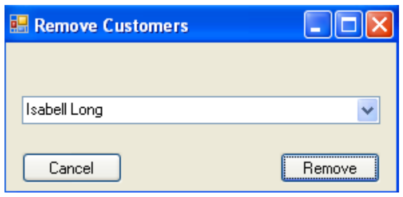
Sometimes, it may be necessary to permanently remove a customer, supplier, or component. To do so, return to the main menu by pressing Close or Quit on several forms, or by logging in again, and click the ‘Remove’ button, or Tab to it and hit Enter. The form below is what is initially displayed.



5.2.6.1 Customers

To remove a customer, select Customer from the drop down menu in ‘frmRemove’, then select the appropriate customer in the form that appears and click ‘Remove’, like in the example for removing the customer ‘Isabell Long’.

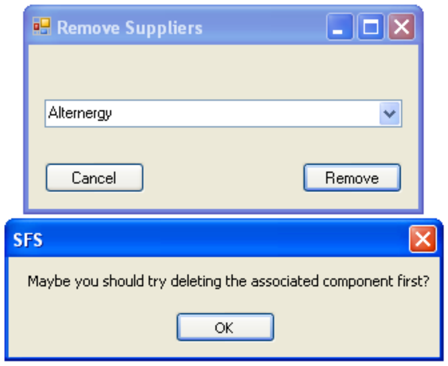




To close the form after successful customer removal, click ‘Cancel’.

5.2.6.2 Suppliers

To remove a supplier, select Supplier from the drop down menu in ‘frmRemove’, then select the appropriate supplier in the form that appears and click ‘Remove’. If the supplier has components associated with it, it will not be removed and an error message will display.

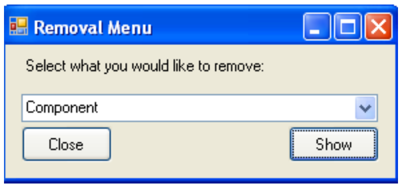


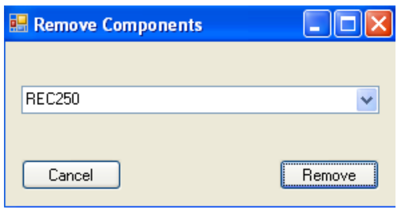
Maybe you should try deleting the associated component first?

To close the form after successful or unsuccessful supplier removal, click ‘Cancel’.

5.2.6.3 Components

To remove a component, select Component from the drop down menu in ‘frmRemove’, then select the appropriate supplier in the form that appears and click ‘Remove’, like in the example for removing the component ‘REC250’.

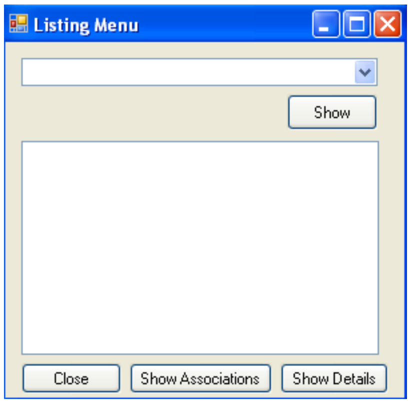




To close the form after successful component removal, click ‘Cancel’.

5.2.7 Listing customers, suppliers, or components

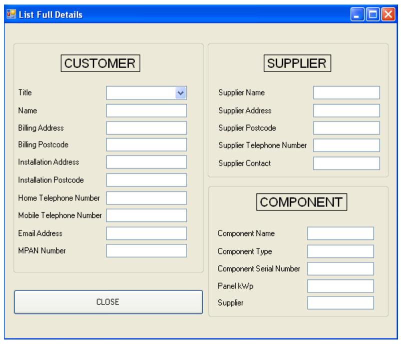
It may be useful to list the contents of the database without having to delve into the database. This can be done by clicking the List button on the main menu. A window as in the screenshot below will appear.



There are many buttons at the bottom of this form: one to close the form, one to list the full details of the selected customer, and one to show any associations between customers, suppliers, or components.

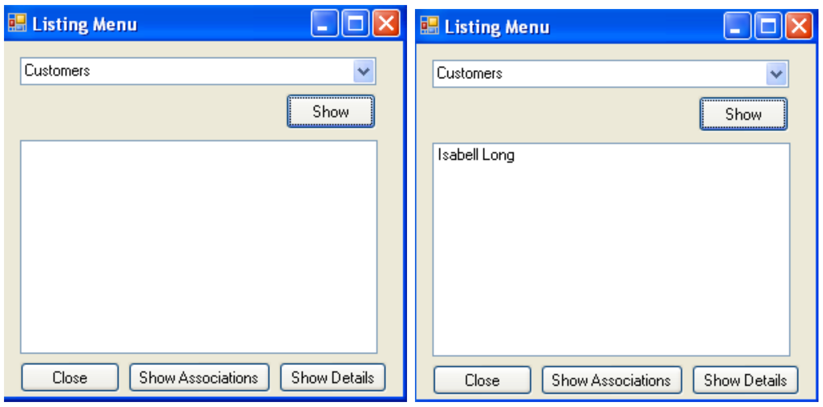
Please note that at present, the show associations button only works when component is selected, as it shows which supplier the component is supplied by.

Upon clicking the button ‘Show Details’ in the List form, a form will display that looks in the first instance like the screenshot below: this is where the full details of the selected customer/supplier/component will be shown. This manual will delve into that a bit later on.



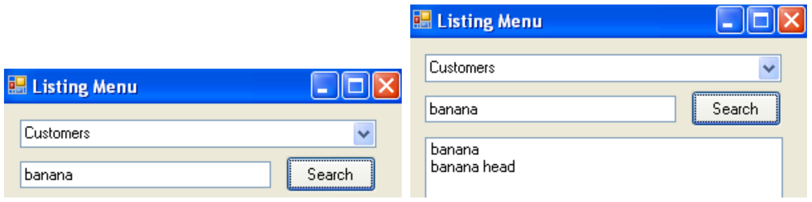
5.2.7.1 Customers

To list customers, select Customers from the dropdown box. Something akin to the following will display[1].



Isabell Long

To search for a customer, enter a search term in the search box screenshoted below, and click Search. The search results will display.



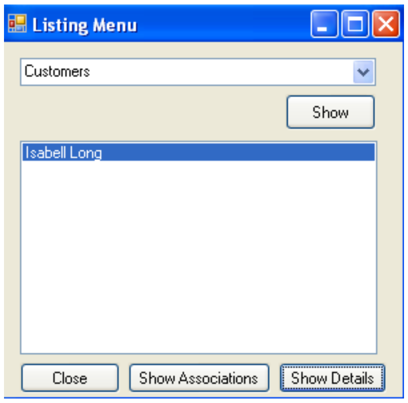
banana

banana

banana
banana head

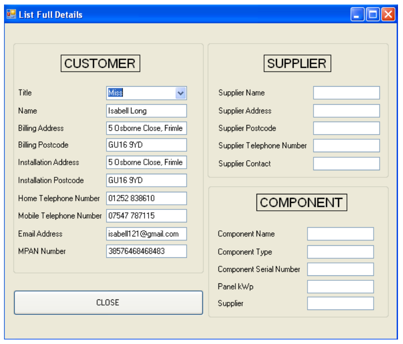
To see the full customer details, click the 'Show Details' button:

[1]In these examples, I have re-added the customer 'Isabell Long' who was deleted earlier in the deletion explanations, for simplicity.



Customers

Isabell Long

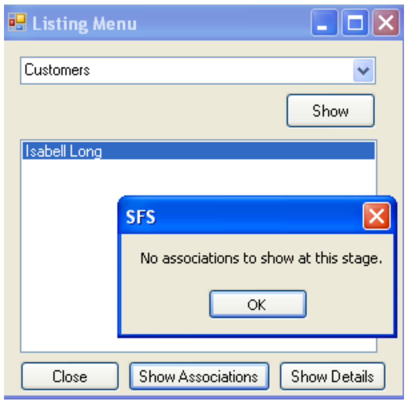


Title	Miss
Name	Isabell Long
Billing Address	5 Osborne Close, Frimle
Billing Postcode	GU16 9YD
Installation Address	5 Osborne Close, Frimle
Installation Postcode	GU16 9YD
Home Telephone Number	01252 838610
Mobile Telephone Number	07547 787115
Email Address	isabell121@gmail.com
MPAN Number	38576468468483

Supplier Name	
Supplier Address	
Supplier Postcode	
Supplier Telephone Number	
Supplier Contact	

Component Name	
Component Type	
Component Serial Number	
Panel kWp	
Supplier	

The 'Show Associations' button will show an error message if it is clicked anywhere but when 'Components' is selected in the dropdown box.



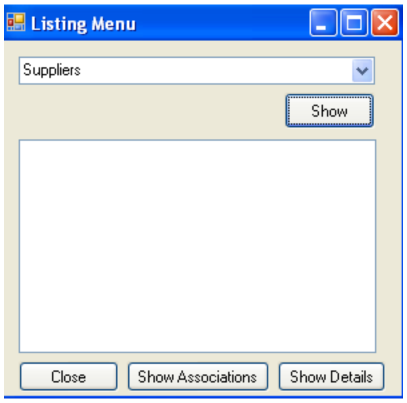
Customers

Isabell Long

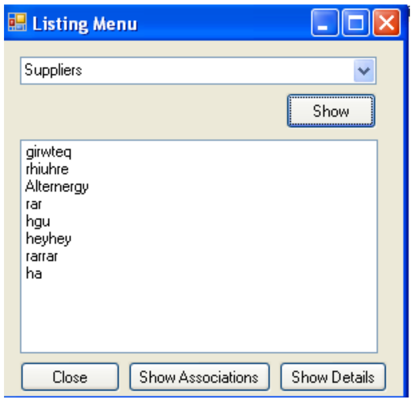
No associations to show at this stage.

5.2.7.2 Suppliers

To list suppliers, select Suppliers from the dropdown box. Something akin to the screenshot below will display[2].



Suppliers



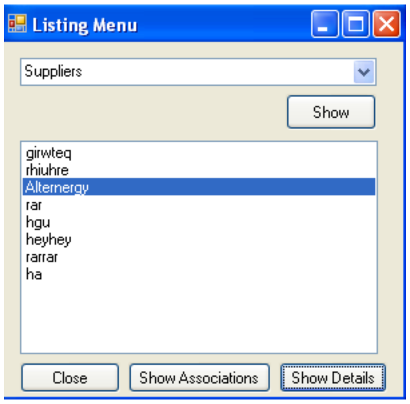
Suppliers

girwteq
rhiuhre
Alternergy
rar
hgu
heyhey
rarrar
ha

To search for a supplier, enter a search term in the search box as screenshotted in the customer listing section. The search results will display.

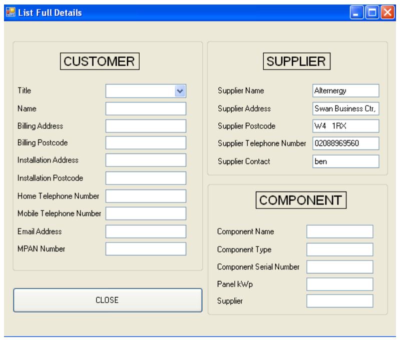
To see the full supplier details, click the 'Show Details' button:

[2]In these examples, I have re-added the supplier 'Alternergy' that was deleted earlier in the deletion explanations, for simplicity.



Suppliers

girwteq
rhiuhre
Alternergy
rar
hgu
heyhey
rarrar
ha

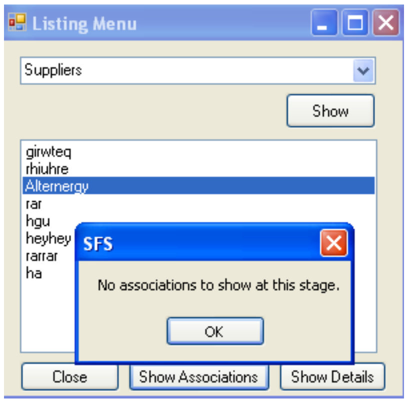


Title	
Name	
Billing Address	
Billing Postcode	
Installation Address	
Installation Postcode	
Home Telephone Number	
Mobile Telephone Number	
Email Address	
MPAN Number	

Supplier Name	Alternergy
Supplier Address	Swan Business Ctr,
Supplier Postcode	W4 1RX
Supplier Telephone Number	02088969560
Supplier Contact	ben

Component Name	
Component Type	
Component Serial Number	
Panel kWp	
Supplier	

The 'Show Associations' button will show an error message if it is clicked anywhere but when 'Components' is selected in the dropdown box.



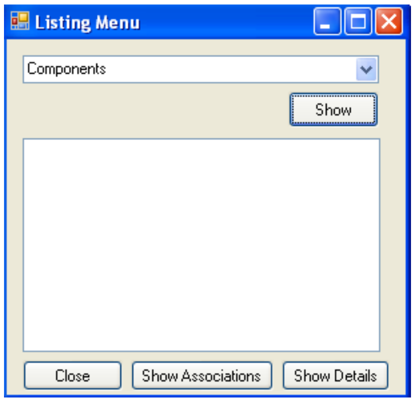
Suppliers

girwteq
rhiuhre
Alternergy
rar
hgu
heyhey
rarrar
ha

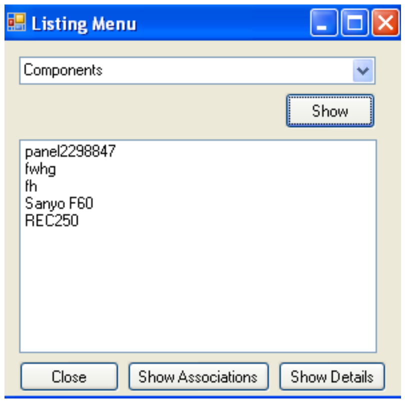
No associations to show at this stage.

5.2.7.3 Components

To list components, select Components from the dropdown box. Something akin to the screenshot below will display.[3]



Components



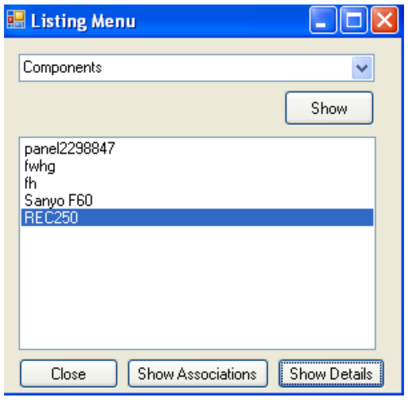
Components

panel2298847
fwhg
fh
Sanyo F60
REC250

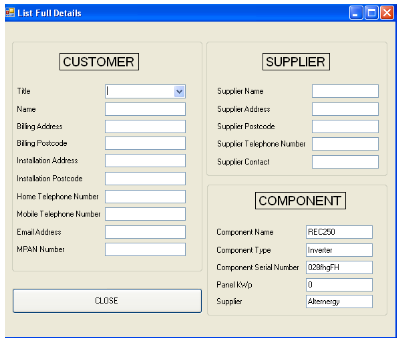
To search for a component, enter a search term in the search box as screenshoted in the customer listing section, and click Search. The search results will display.

[3]In these examples, I have re-added the component 'REC250' that was deleted earlier in the deletion explanations, for simplicity.

To see the full component details, click the ‘Show Details’ button.

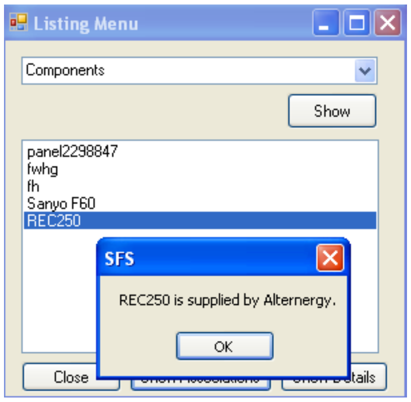


panel2298847
fwhg
fh
Sanyo F60
REC250



Title	
Name	
Billing Address	
Billing Postcode	
Installation Address	
Installation Postcode	
Home Telephone Number	
Mobile Telephone Number	
Email Address	
MPAN Number	
Supplier Name	
Supplier Address	
Supplier Postcode	
Supplier Telephone Number	
Supplier Contact	
Component Name	REC250
Component Type	Inverter
Component Serial Number	028fhgFH
Panel kWp	0
Supplier	Alternergy

Now the ‘Show Associations’ button will not display an error message! Clicking that button when the selected component is ‘REC250’ will display that the component is supplied by ‘Alternergy’, and so on.

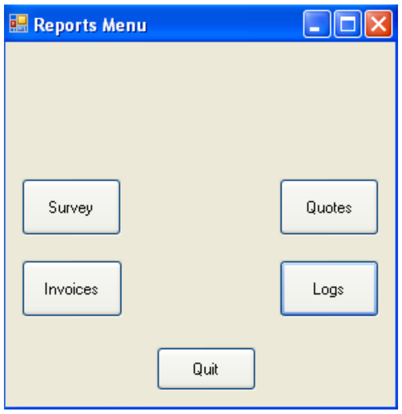


panel2298847
fwhg
fh
Sanyo F60
REC250

REC250 is supplied by Alternergy.

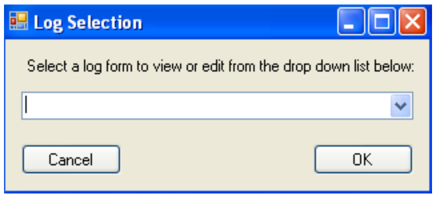
5.2.8 Reports

Reports can be produced for various things using this system: logs, quotes, invoices, and surveys. To access the reports menu, click Reports from the main menu; the form shown should appear.

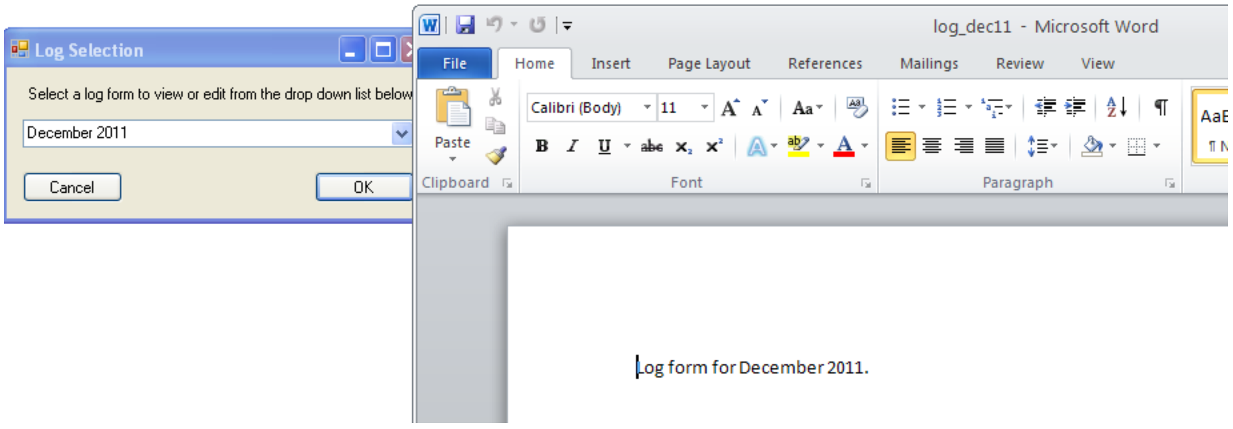


5.2.8.1 The log forms

Clicking ‘Logs’ will open a screen like the one below.


Log Selection
Select a log form to view or edit from the drop down list below:
Cancel
OK

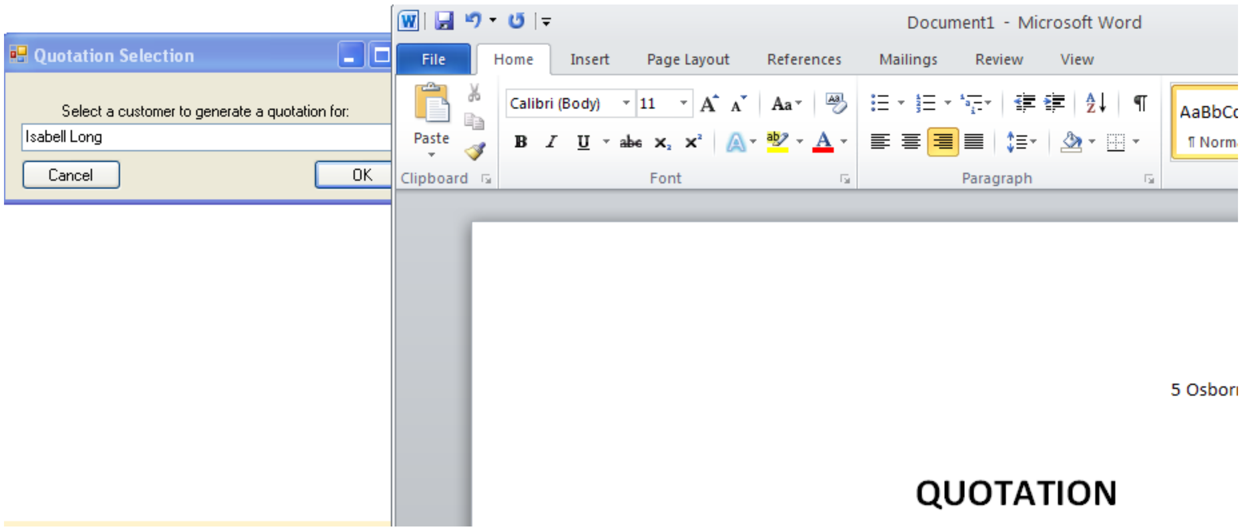
Select a month/year from the drop down menu, and click OK. This will open the log form for the selected month and year in Microsoft Word, which you can edit and save. Close the Word document to return to the program, or just click the program window on the task bar. See the screenshots for an example of the log form from December 2011.



Log form for December 2011.

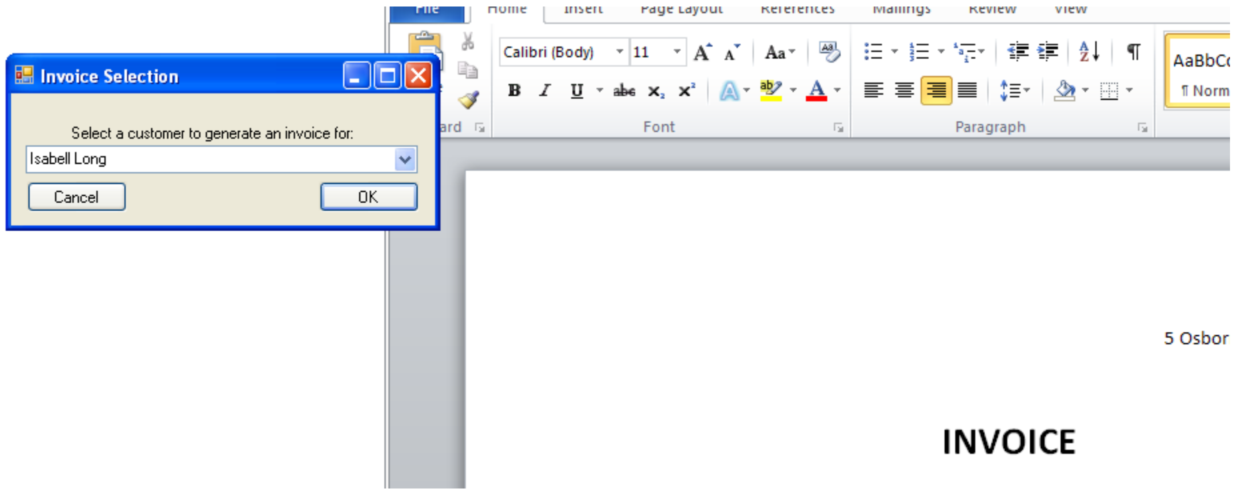
5.2.8.2 The quotations

Currently, the quotations only automate inputting of the billing address, however they are still useful. To open a quotation in Microsoft Word, click ‘Quotes’ in the report menu and select a customer to open the quote for. An example is below.



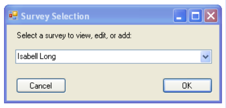
5.2.8.3 The invoices

Currently, the invoices only automate inputting of the billing address, however they are still useful. To open an invoice in Microsoft Word, click 'Invoices' in the report menu and select a customer to open the invoice for.



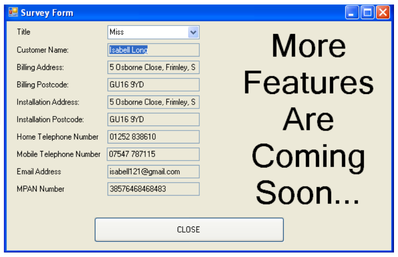
5.2.8.4 The surveys

Currently, the surveys only display the customer details, and no other details can be added. To view the surveys, click the 'Surveys' button in the report menu and select a customer to open the survey for.



Select a survey to view, edit, or add:

Isabell Long



Title	Miss
Customer Name:	Isabell Long
Billing Address:	5 Osborne Close, Frimley, S
Billing Postcode:	GU16 9YD
Installation Address:	5 Osborne Close, Frimley, S
Installation Postcode:	GU16 9YD
Home Telephone Number	01252 838610
Mobile Telephone Number	07547 787115
Email Address	isabell121@gmail.com
MPAN Number	38576468468483

5.2.9 Error handling

Error text	Recovery
"Incorrect username or password. Try again."	An incorrect username or password, or both, has been entered. Click 'OK' and re-enter the login details.
"Input a valid email address!"	You most likely did not include an @ symbol in the entered email address—maybe you were on a US keyboard and entered a " instead? Click 'OK' and reenter the email address in its entirety, properly.
"There has been an error."	One or more of the textboxes/ dropdowns were blank, most likely. Click OK and make sure that all the textboxes have something in them, even 'NULL' for on-purpose values that do not have information for, then click Save again.
"This window will close and these details will not be saved."	Self-explanatory. Click the 'OK' button to close the window without saving the details that may or may not have been entered.
"Enter a value in each of the boxes, and make it a valid one!"	Not all the textboxes have been populated with data. Populate them, even with "NULL", then click 'Save' again.
"No associations to show at this stage."	Informational. Click 'OK' to dismiss the box.
"Cannot show associations; please select an item from the list."	The program cannot show the associated supplier for the component because no component has been selected. Click 'OK' to dismiss this box, select a supplier, then reclick 'Show Associations'.
"Select a customer to generate an invoice for!"	This appears because no customer has been selected from the dropdown box at the top of the form from whose details an invoice can be generated. Click the dropdown box and select a customer, after having dismissed the error message, then redisplay the invoice.

Error text (cont.)	Recovery (cont.)
"No form was selected."	No log form was selected from the drop-down box. Click 'OK' to dismiss the error message, then actually select a log form to open from the list and click 'OK' on the selection form.
"Data insertion failed."	Self-explanatory. Click 'OK' to dismiss the error and re-enter whichever information was not included or was in the wrong format.

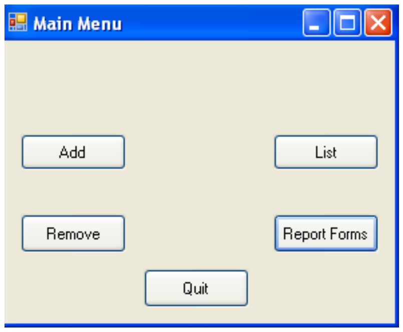
Chapter 6

Appraisal

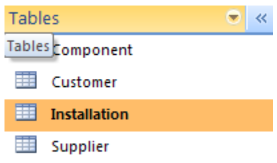
6.1 Comparison of project performance against numbered general and specific objectives

My objectives from Section 1 were:

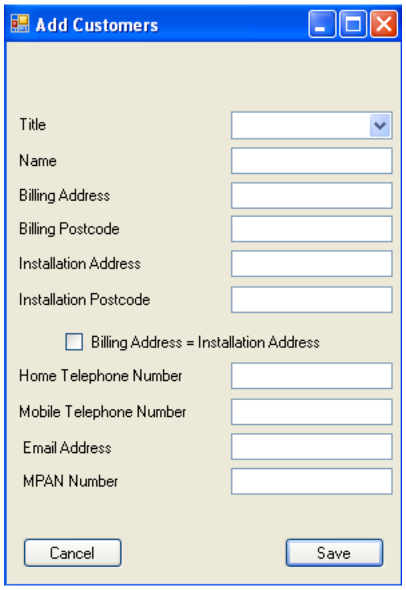
1. Have a main menu that allows the user to select different options.



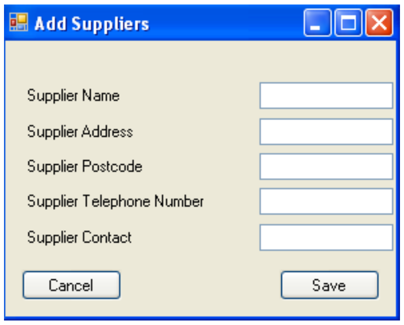
2. Have a functioning relational database, queriable with SQL, made in Microsoft Access, with the required number of tables.



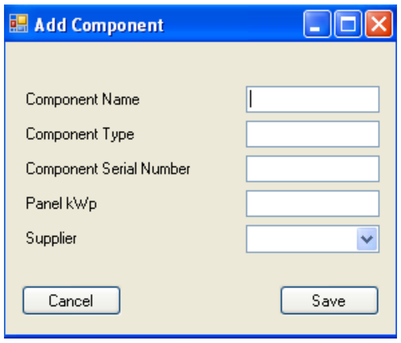
3. Enable the user to add customers.



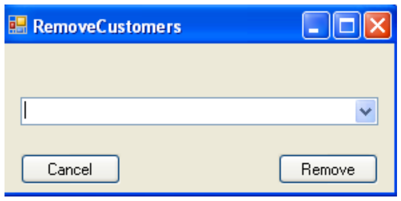
4. Enable the user to add suppliers.



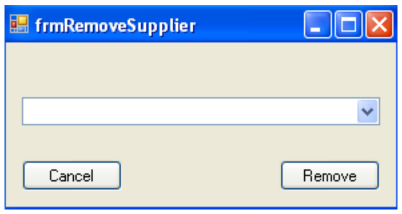
5. Enable the user to add components.



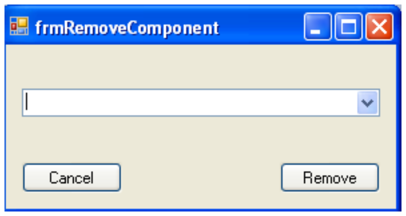
6. Enable the user to remove customers.



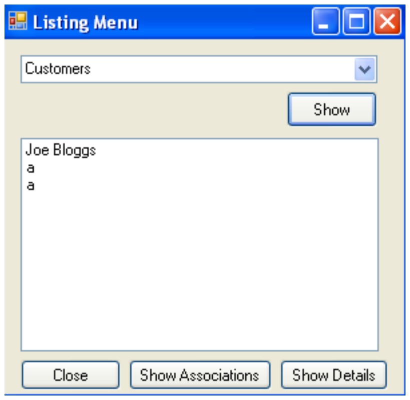
7. Enable the user to remove suppliers.



8. Enable the user to remove components.



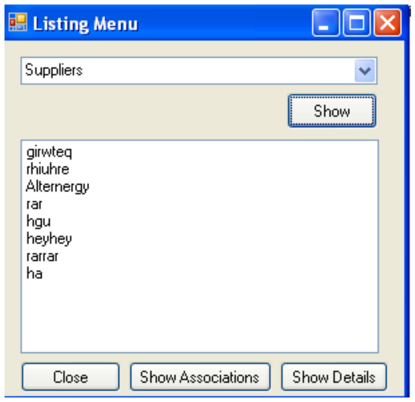
9. Enable the user to list customers.



Customers

```
Joe Bloggs
a
a
```

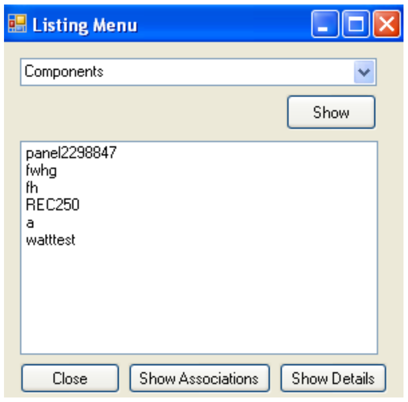
10. Enable the user to list suppliers.



Suppliers

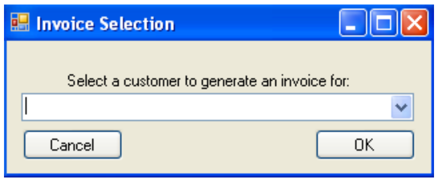
```
girwteq
rhiuhre
Alternergy
rar
hgu
heyhey
rarrar
ha
```

11. Enable the user to list components.



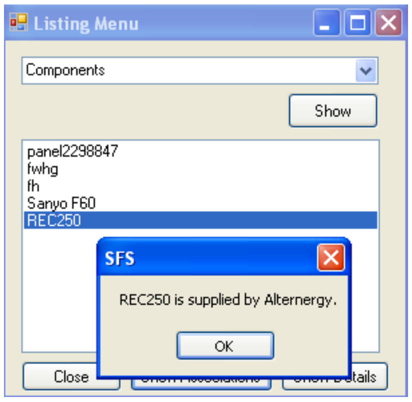
panel2298847
fwhg
fh
REC250
a
watttest

12. Enable the user to view and edit invoices, log forms, and reports in Microsoft Word by clicking buttons in the program to open the requested forms.



Select a customer to generate an invoice for:

13. Enable the user to view relationships between suppliers and the components they stock.

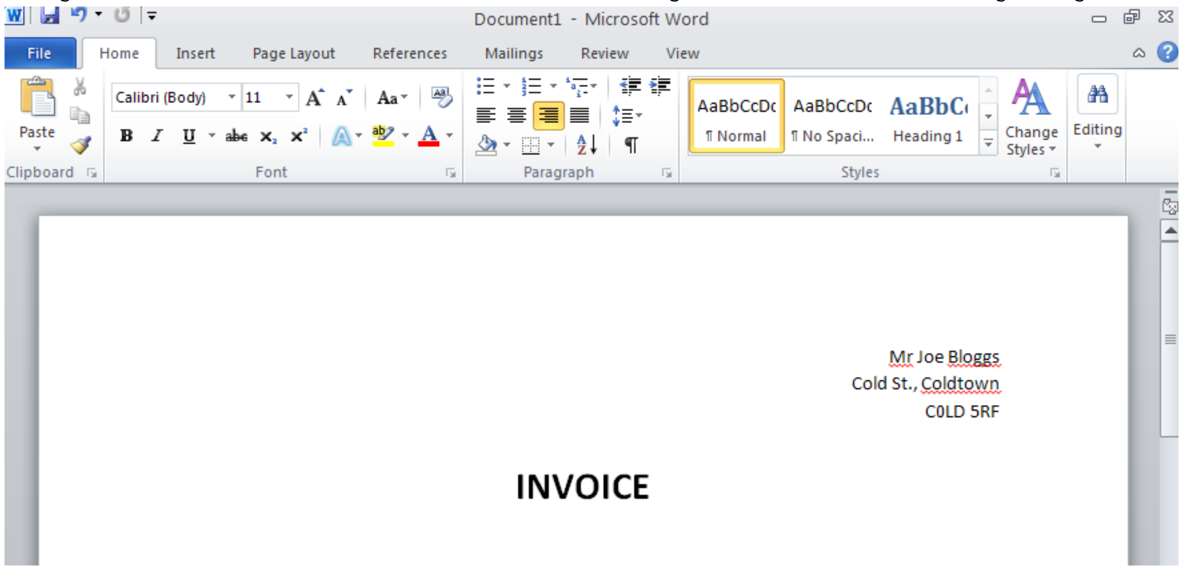


panel2298847
fwhg
fh
Sanyo F60
REC250

REC250 is supplied by Alternergy.

14. The program must input data into the invoices and quotes to avoid the user having to duplicate data entry, and this data must come from either user input or data from the database.

- Objective fourteen is partially acheived due to the invoices being populated with the chosen customer's address, and a header 'INVOICE'.



Mr Joe Bloggs
Cold St., Coldtown
COLD 5RF

INVOICE

15. The system should be menu-driven, with consistent GUI form layout throughout, as far as possible.

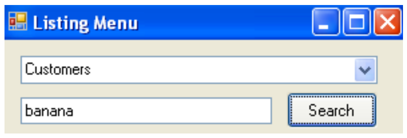
    - See all of the forms for evidence of this: they are menus.

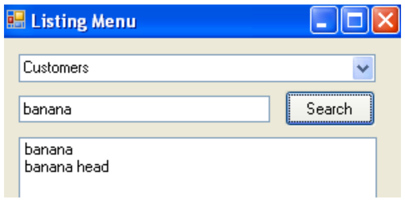
16. Enable printing directly from the program for the user to print lists of customers etc.

    - This has not been acheived due to the unexpected complexity of printing to an actual printer in VB.NET. Luckily, this objective was not strictly necessary due to the reports being able to be printed via Microsoft Word when they opened.

17. Enable searching of customers, suppliers and components, and sort the search results.

    - This objective is partially achieved: the sorting of the search results was not implemented due to time constraints.

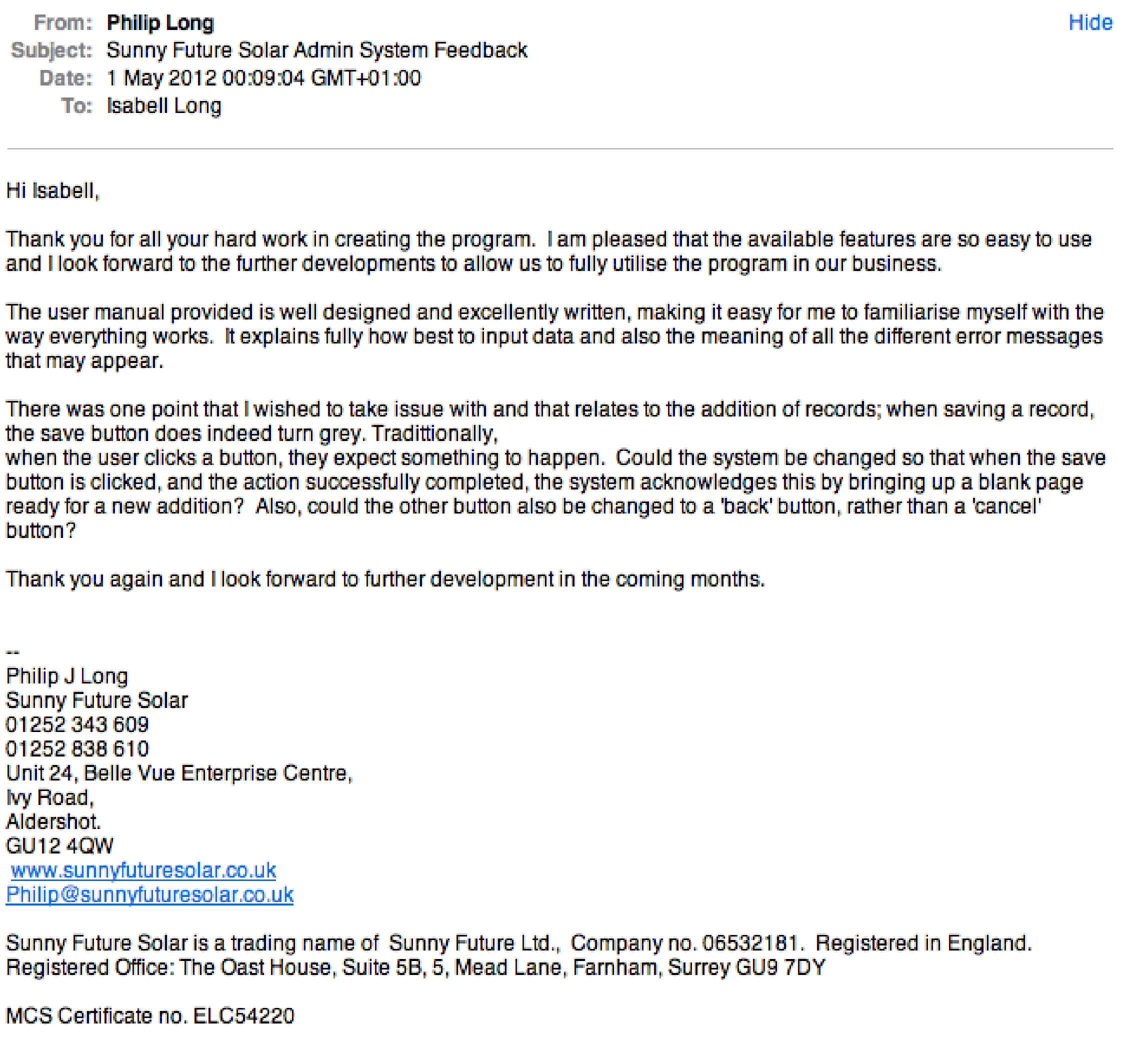




banana
banana head

I have acheived objectives one, two, three, four, five, six, seven, eight, nine, ten, eleven, twelve, thirteen and fifteen in full. These objectives are referenced clearly in the code comments, with screenshots above in the bulleted list.

6.2 User feedback authenticated by the assessor



From: **Philip Long**
Subject: Sunny Future Solar Admin System Feedback
Date: 1 May 2012 00:09:04 GMT+01:00
To: Isabell Long

Hide

Hi Isabell,

Thank you for all your hard work in creating the program. I am pleased that the available features are so easy to use and I look forward to the further developments to allow us to fully utilise the program in our business.

The user manual provided is well designed and excellently written, making it easy for me to familiarise myself with the way everything works. It explains fully how best to input data and also the meaning of all the different error messages that may appear.

There was one point that I wished to take issue with and that relates to the addition of records; when saving a record, the save button does indeed turn grey. Tradittionally,
when the user clicks a button, they expect something to happen. Could the system be changed so that when the save button is clicked, and the action successfully completed, the system acknowledges this by bringing up a blank page ready for a new addition? Also, could the other button also be changed to a 'back' button, rather than a 'cancel' button?

Thank you again and I look forward to further development in the coming months.

--
Philip J Long
Sunny Future Solar
01252 343 609
01252 838 610
Unit 24, Belle Vue Enterprise Centre,
Ivy Road,
Aldershot.
GU12 4QW
www.sunnyfuturesolar.co.uk
Philip@sunnyfuturesolar.co.uk

Sunny Future Solar is a trading name of Sunny Future Ltd., Company no. 06532181. Registered in England.
Registered Office: The Oast House, Suite 5B, 5, Mead Lane, Farnham, Surrey GU9 7DY

MCS Certificate no. ELC54220

6.3 Analysis of user feedback

- Philip liked the user manual and its layout and the explanation of error messages.
- He "looked forward to further developments", of which there are a few that he picked up on:
  - The user interface needs some work with regard to the add customer/supplier/component forms, with it at present not telling the user when a customer etc. has been successfully inserted and not leading them back to the form to enter another customer. Philip's suggestion is possible to implement with a loop: if there are no errors then insert into the database and, if successful, loop back around to the empty 'Add[Customer|Supplier|Component] form for ease of data entry.

– Also UI related, Philip requested that the Cancel button on each of the addition forms be changed to a Close button, because then it does not give the impression that the user is eradicating a legitimate change, or has made one even when the form is blank. This is possible through simply changing a few procedure and variable names and references.

6.4 Possible extensions

Lots could be done to extend the feature set of this project, including fixes for the issues raised in the user feedback, and:

- Displaying invoices and quotations within the program itself, using a report viewer integrated into VB, not having to go through Microsoft Word, enabling the information to be displayed more quickly and reducing the amount of times the user has to click.
- Enabling printing of the aforementioned invoices and quotations from within VB. This would be faster as the computer would not have to load a word processor and the user would not have to click three buttons—only one.
- Enabling the invoices and quotations to display everything they should display: the selected components going to be installed and installation price to name but a few things.
- Finishing development of the survey form.
- Rewriting the program as a console application in Ruby as another learning experience? This would make the program cross-platform and—with the absence of buttons and graphics—potentially more simple.

Chapter 7

References

7.1 VB.Net

- Practical Database Programming with Visual Basic.NET – Ying Bai – ISBN: 9780521712354.
- Microsoft's MSDN library – `http://msdn.microsoft.com/en-us/library/default.aspx`.

7.2 Software

- Microsoft Office Access.
- Microsoft Office Word.
- Visual Studio 2010.

7.3 3rd party code libraries

- System.Data.
- System.Data.OleDb.
- Microsoft.Office.Core.
- Microsoft.Office.Interop.
- Microsoft.Office.Word.