

Fall 12-1-2006

Order Processing System

Sandeep Modugu
Dakota State University

Follow this and additional works at: <https://scholar.dsu.edu/theses>

Recommended Citation

Modugu, Sandeep, "Order Processing System" (2006). *Masters Theses*. 113.
<https://scholar.dsu.edu/theses/113>

This Thesis is brought to you for free and open access by Beadle Scholar. It has been accepted for inclusion in Masters Theses by an authorized administrator of Beadle Scholar. For more information, please contact repository@dsu.edu.



PROJECT APPROVAL FORM

We certify that we have read this project and that, in our opinion, it is satisfactory in scope and quality as a project for the degree of Master of Science in Information Systems.

Student Name: Sandeep Modugu

Master's Project Title: Order Processing System

Faculty supervisor: Wm Shan Date: 12/4/06

Committee member: Mark Moran Date: 12/4/06

Committee member: Stephen Gelsley Date: 12/4/06

Order Processing System

**A report submitted in partial fulfillment of the requirements
for the degree of**

**Master of Science in Information Systems
Dakota State University**

2006

By

Sandeep Modugu

Project Committee:

Ronghua Shan, Ph.D., Faculty Mentor and Chair

Mark Moran, Ph.D., Committee Member

Stephen Krebsbach, Ph.D., Committee Member

ACKNOWLEDGMENT

First I would like to thank committee members, Prof. Ronghua Shan, Prof. Mark Moran and Prof. Stephen Krebsbach who encouraged me, communicated confidence in my chosen direction and corrected my efforts and understanding. Without their knowledge, I would have had a much more difficult time completing this project.

I will like to thank all MSIS faculty members who have one way or the other contributed toward my study at Dakota State University. I would also like to appreciate the support provided by my family. I thank them for their support and steadfast love.

ABSTRACT

Developing business demands high technology needs for performing its basic operations. Processing of orders and maintaining consistent data has been the major focus through years. This report is a mere representation of a system, which allows a high-end Order Processing System. Maintaining consistent data, ease of use system for managing and recording transactions and transformation of incoming order flows has been the major focus behind its development. This has been accomplished using tools provided by the .Net Framework such as ADO .Net for talking to database and ASP .Net for processing electronic orders, both using C# language. Thus, we have incorporated a system that not only process orders from customers online but also provide the managing staff with significant front ends so as to make their tasks easier and faster. This system solves the major problem of inconsistency, inaccurate throughput of orders and provides the company with reliable and easy to use system.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Objective	1
1.2 Scope	1

2. PROBLEM STATEMENT AND ANALYSIS

2.1 Problem Area	2.
2.2 Problem Analysis	2
2.3 Solution to the problem	3

3. SYSTEM ANALYSIS

3.1 Functional Requirements	
3.1.1 Management Requirements.....	4
3.1.2 Customer Requirements.....	5
3.2 Specific Requirements	
3.2.1 External Interface Requirements.....	6
3.2.2 Validation of data.....	6
3.3 Performance Requirements.....	
3.3.1 Performance.....	6
3.3.2 Security.....	7
3.3.3 Software validation.....	7

4. DESIGN

4.1 User Manual.....	8
4.2 System Design	
4.2.1 Three- tier Architecture.....	14
4.2.2 About Microsoft .Net Framework	16
4.2.3 Why .Net?	18
4.3 Database Design	19
4.4 System Design	24

5. IMPLEMENTING THE SYSTEM

5.1 At Customer side	28
5.2 At Management side	30
5.3 Black Box Testing	34

6. CONCLUSIONS

6.1 Project Enhancements35

Appendices

User Interfaces

Source Code

Database Details

References

LIST OF FIGURES

Figure1: Home Page.....	8
Figure2: Orders Page.....	9
Figure3: Order processing main window.....	10
Figure4: New Orders Page.....	11
Figure5: Check Inventory Form.....	12
Figure6: New Inventory Form.....	13
Figure7: Three tier Architecture.....	14
Figure8: Data base Design.....	19
Figure 9: The explanation for use case diagram.....	26
Figure10: Login Form.....	36
Figure11: Order Process.....	37
Figure12: New Order Page.....	38
Figure13: New Inventory Page.....	38
Figure14: Order Status.....	39
Figure 15: Inventory Search Page.....	40

1. INTRODUCTION

1.1 Objective

The aim of this project is to provide the coffee giants Beans'r'us with a system which can perform all the major operations of order processing. On one hand the system should be capable of serving online customers, and on the other it should allow an easier way to place and retrieve order information.

1.2 Scope

The major focus of this project is to provide users with a system that not only solves business problems but also can maintain consistency, atomicity and accuracy. The system thus developed can provide customers with a well defined and easy to use interface to place orders. On the other hand, the significant task of order processing, is performed and maintained in more simplified manner, which includes accepting orders, checking inventory levels, generating invoice, shipping and delivery, updating accounts and finally figuring them graphically. All of these tasks are accomplished keeping in mind the user's capability.

2. PROBLEM STATEMENT & LIMITATIONS

2.1 Problem Area

In recent years, processing customer orders has been of much significance. Expanding business' demands sophisticated infrastructure through which, this process can be achieved up to expectations. In this project, we deal with an upcoming company Beans'r'us dealing in coffee products. The company has established its roots in the market and thus demands better infrastructure to process, store and maintain orders. It needs a system which can provide customer's with simplified user interfaces, thus placing orders. On the management side, it needs a system which can track the orders and process them more effectively than the current one.

2.2 Problem Analysis

Going through the company's working strategy; we take into consideration the following:

- i. Most of the company's order processing is carried on by Terry – the order processor.
- ii. Apart from the simple and well structured user interface we should take into Consideration Terry's working ability, for which she says as "I work mostly with the keyboard since a mouse really slows me down. Fortunately, our system is all form fields so I can enter the information in the fields as possible".
- iii. Terry takes care of everything starting from placement of order till its shipment and accounts; which includes checking inventory, ordering inventory, evaluating order data, generating invoice, managing accounts and presenting reports. Thus a system which can perform all these tasks effectively with an easy navigation would serve her for better results.
- iv. Faster retrieval of information from the database needs to be done while providing details such as Order status, Inventory levels etc.

- v. Orders are placed either electronically online or over phone or through sales representative forms. Thus an elegant discrimination of presenting and working needs to be implied accordingly.
- vi. Going a level up, we need data such as Inventory Reports and Sales report which are to be presented graphically to the company CEO, Rocky.

2.3 Solution to the problem:

We would finally sort out the solution which perhaps solves most of the raised facts during analysis. The following list illustrates them respectively:

- i. To make Terry work effectively, we need to consider her capability of working with keyboard and thus we need an interface with lots of keyboard shortcuts for navigation and working.
- ii. A system with simple user forms, for entering and retrieving order information, needs to be designed.
- iii. Transactions with the customers and suppliers are to be maintained distinctly to avoid information chaos.
- iv. Periodical information updates and cancellations is to be allowed so as to maintain data consistency.
- v. Each order is to be identified distinctly by using its ID with its corresponding data values. The same would apply for the customer and the supplier. This would allow accurate data retrieval from all departments.
- vi. Graphical tools are needed for presentation, which makes it easier for the higher management levels to compare and record improvements.

3. SYSTEM ANALYSIS

3.1 Functional Requirements.

3.1.1. Management Requirements:

The requirements at the management side can be categorized as follows:

i. Authentication

The system at the management side should be restricted to specific users, which have their own unique username and its corresponding password. Unauthorized users should not get an access to the systems working.

ii. New Orders

New orders should be entered and maintained effectively by the system. Each order thus generated should contain a unique identifier Order ID that should be assigned by the system itself, which allows distinction between different orders. Identification of orders is mostly done through this unique number and thus it is necessary for the user to ensure that correct and relevant information has been stored with its proper Order ID.

iii. Inventory

The system should allow users to check inventory levels and enter new inventory data on purchase. A clear demarcation between purchasing existing items and purchasing new items should be made. Both of which should be tracked with existing and new suppliers respectively.

iv. Tracking Orders

Efficient way of retrieving the order status should be possible with the system. Data thus retrieved should be consistent and updated.

v. Avoiding duplication

The system should keep a record of reoccurring information such as customer, supplier and items so as to make the working of the system faster.

vi. Record generation

The system should be capable of generating graphical figures in order to track the company's improvement.

3.1.2 Customer Requirements

i. Identification

The system will maintain a record of new and old customers. Hence it is necessary for the customers to record their own ID, which serves for future reference.

ii. Consistency

It is necessary for every customer to ensure that the data which he/she provides is accurate while placing orders.

iii. Easy Navigation

Good navigation techniques and an ease of use interface has to be presented to the customers in order to make the ordering system more effective.

iv. Concurrency

Simultaneous transactions should be avoided for data consistency and integrity

3.2 Specific Requirements

3.2.1 External Interface Requirements

Graphical user interface

The system will be using a graphical user interface, as it is important that all work related tasks, which are supported by the system, can be handled fast and efficiently.

Error messages should be easy to understand and are to be structured in a consistent manner. The system should be structured in such a way that an error message will never leave the employee in a deadlock.

3.2.2 Validation of Data

This function will not be visible to the users. It will ensure that the data entered into the system is in the correct format, which again will ensure that the system works properly.

The data sent to the server will be validated in the user interface. If the user tries to save invalid data he/she will get an error message describing the problem. Thus only valid data will be entered into the database.

3.3 Performance Requirements

3.3.1 Performance

The system should consist of a web-interface linked to a database. It should be fast in all aspects: Database manipulation and when filling in the forms. The database performance depends on the use of a database server and also on which type of server Beans'r'us used (due to the fact that some servers offer the possibility to use query optimization and more indexing facilities than others). A Relational Database System will be used to improve the Database performance and indexing will be used to make the database queries faster. To improve the front-end performance, different techniques will be utilized to make the system faster and more efficient (e.g. popup windows and dropdown lists), making less space for errors.

3.3.2 Security

The system will be able to support multiple users at the same time, by means of the built in facilities in the database system. We hold no responsibility regarding backup and IT-security. The Beans'r'us system administration is liable for these operational aspects.

3.3.3 Validation

Software validation will be performed prior to the scheduled release, to ensure conformity between the requirements. It will be done by a series of different tests. These tests will be carried out according to a verification and validation test plan which contains specific test cases to ensure that all requirements are satisfied.

4. DESIGN

4.1 User Manual

4.1.1 Introduction

This section is a guide to the users who are using the Beans'r'us system. This section is sub categorized into the workings at the Customer's side and the workings at the Management's side.

4.1.2 System at the Customer's side

As soon as the user logs onto the Beans'r'us website, he is presented with an interface wherein can choose to navigate to the products page to view all the available products. The interface looks as follows:

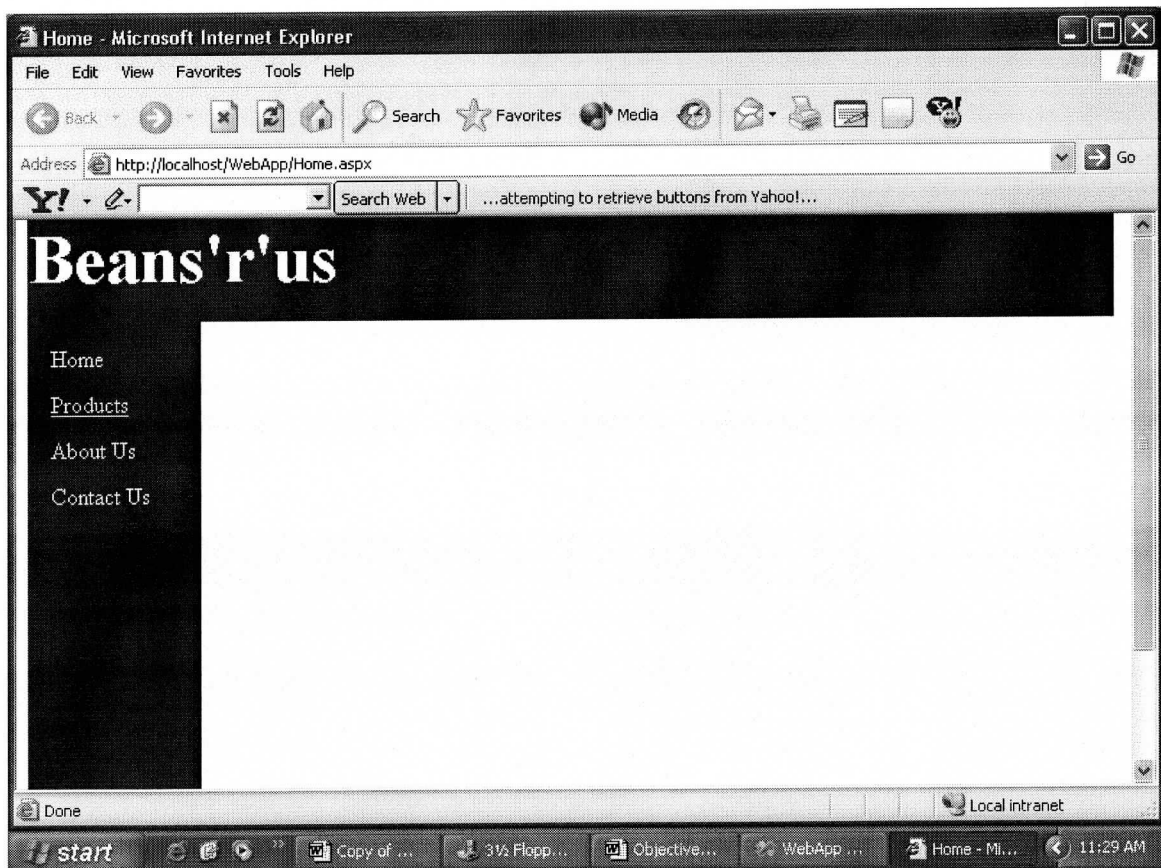


Figure1: Home Page

To buy a product, just check the box present beside it and enter the quantity. You can select one or more products at a time. To order a product press 'Buy' button present at the end of the page.

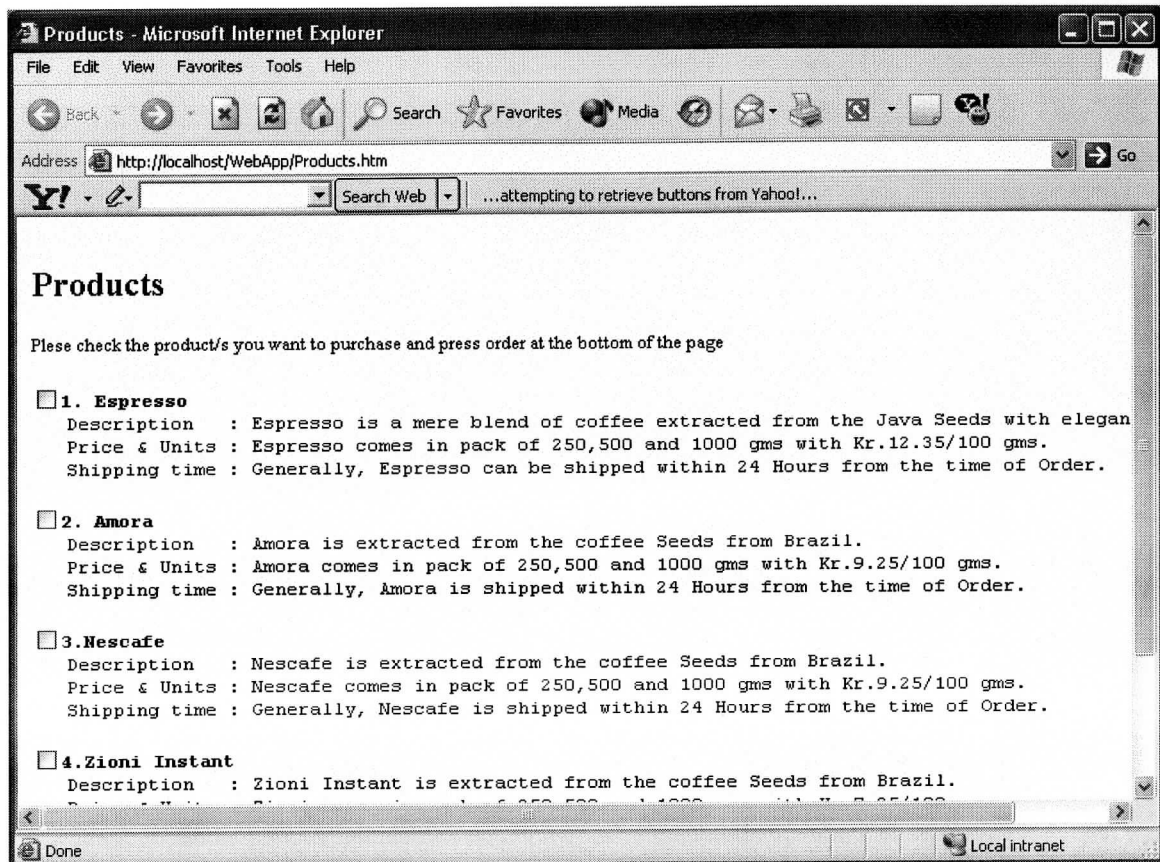


Figure 2: Products Page

You will be prompted to enter your information such as Name, Address etc. If you are an existing customer, just entering your customer ID would be sufficient to place an order, if not, and then you need to enter all the fields. Please recheck your information and the ordered products. Press the button 'Order' to place the order.

Finally, you will be forwarded with your order Id. Make a note of that, all the future reference to your order information will be carried out with your order ID. This finishes your process of order. Sit back and relax while we process your order and send you with further details.

4.1.3 System at Management's side:

This section presents you with working of Beans'r'us system:

i. Order Processing (Main Window):

On its start, the main window termed as the Order Processing window, appears. This window acts as the mother window for all the other forms. This window is loaded with shortcut keys so as to open different forms to perform different operations.

S.No	Tasks	Shortcut Key
1	Open menu	Alt + O
2	New Order Entry	Alt + O + N (or) Ctrl + N
3	Inventory Management	Alt + O + I (or) Ctrl + I
4	Check Inventory	Ctrl + I + C
5	New Inventory	Ctrl + I + Ins
6	Order Status	Alt + O + S (or) Ctrl + S
7	Help menu	Alt + H
8	Help topics	Alt + H + E (or) Ctl + E
9	Search for topic	Alt + H + F (or) Ctrl +F
10	Contact Us	Alt + H + U (or) Ctrl +U

Figure 3 : Order Processing main window

ii. New Order :

The new order window allows to enter the information concerning the order to be placed. The system generates a unique ID for every order placed, which serves as the basis for identifying the orders. Every other data concerning the orders is to be entered. The form also gathers some of the customer information which is entered if it is a new customer. Incase of an existing customer, his/her customer ID is sufficient for entering the order.

To save the order data, press the 'order' button present below the form.

View of New Orders Form.

Products	Qty/Kg		Qty/Kg
<input type="checkbox"/> Espresso	<input type="text"/>	<input type="checkbox"/> Nescafe	<input type="text"/>
<input type="checkbox"/> Capuccino	<input type="text"/>	<input type="checkbox"/> Bru Instant	<input type="text"/>
<input type="checkbox"/> Real Tas	<input type="text"/>	<input type="checkbox"/> Black Cafe	<input type="text"/>
<input type="checkbox"/> Brazil Mix	<input type="text"/>	<input type="checkbox"/> Lipton	<input type="text"/>
<input type="checkbox"/> Amora	<input type="text"/>	<input type="checkbox"/> Zionl Instant	<input type="text"/>

Figure 4: New Orders Page

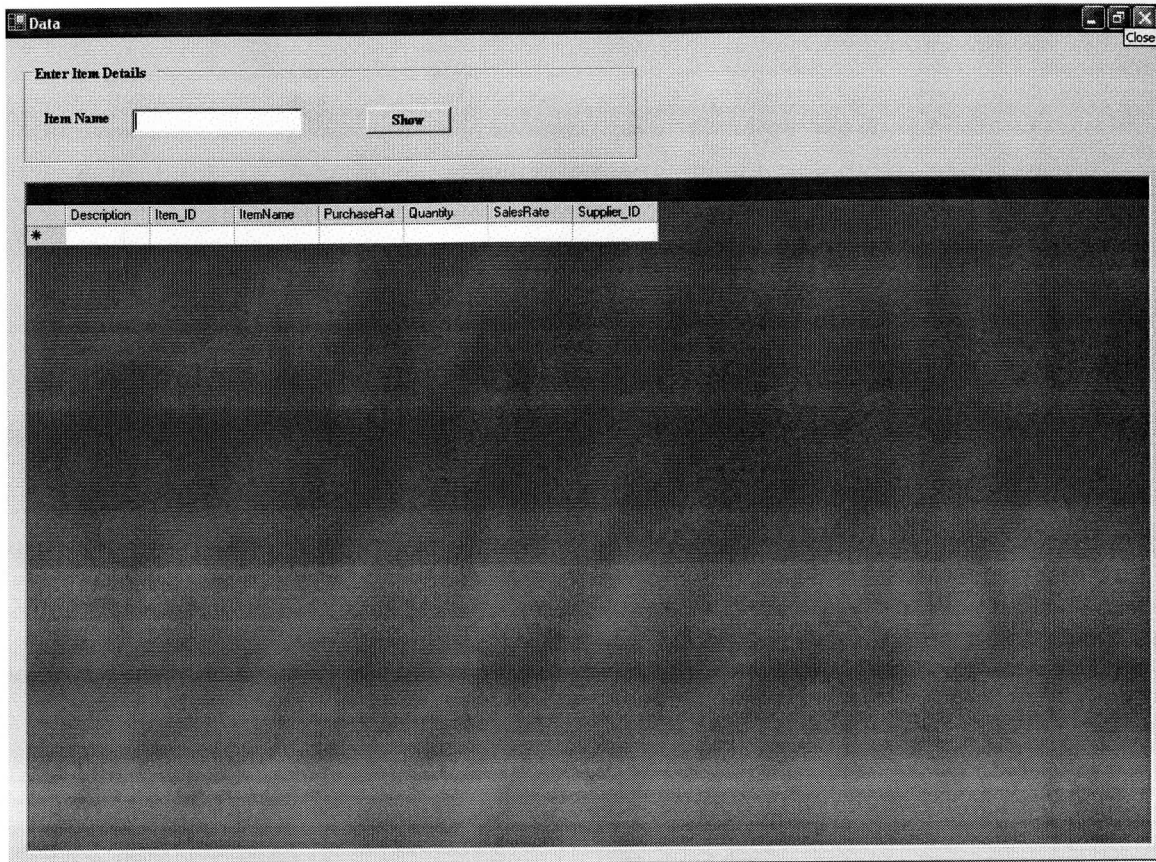
iii. Inventory Management.

Inventory Management allows the user for entering and viewing the inventory levels

Check Inventory Form

This form accepts an input from the user in the form of an item and traverses its corresponding value in the database. If found the result is a corresponding row of the respective table.

View of CheckInventory Form.



The screenshot shows a window titled "Data" with a "Close" button in the top right corner. The main area is titled "Enter Item Details" and contains a text input field labeled "Item Name" and a "Show" button. Below this is a table with the following columns: Description, Item_ID, ItemName, PurchaseRate, Quantity, SalesRate, and Supplier_ID. The table contains a single row with an asterisk (*) in the first cell, indicating a search result. The rest of the table area is dark and mostly obscured.

Description	Item_ID	ItemName	PurchaseRate	Quantity	SalesRate	Supplier_ID
*						

Figure 5: Check Inventory Form

New Inventory Form

This form allows to user to enter information regarding the purchase of a new item. It includes the purchase of an existing item and also the purchase of a new item. To order an existing item, select the item name from the drop down list and press the button next to it.

View of New Inventory Form.

Order Process - [New Inventory]

Open Help

Item Details

Existing Items:

Item ID:

Item Name: Quantity:

Order Date: Required Date:

Purchase Rate: Sales Rate:

Description:

Supplier Information

Existing Supplier New Supplier

Supplier Detail

Address	Email	Phone Num	Supplier ID	SupplierNa
*				

Figure 6: New Inventory Form

iii. Order Status

This form allows checking the current status of the order placed. To check the order status, you need to enter the order ID and then press the button 'Show'. Data concerning to the order is shown in the table below, wherein, order related information such as the order date, shipping data, payment etc is mentioned. The field State informs the user with the current state of the order.

iii. Help Menu

The help menu provides the users with the topics concerning the usage of this system. Users can use the help topics for future reference and also contact us in case of any problems.

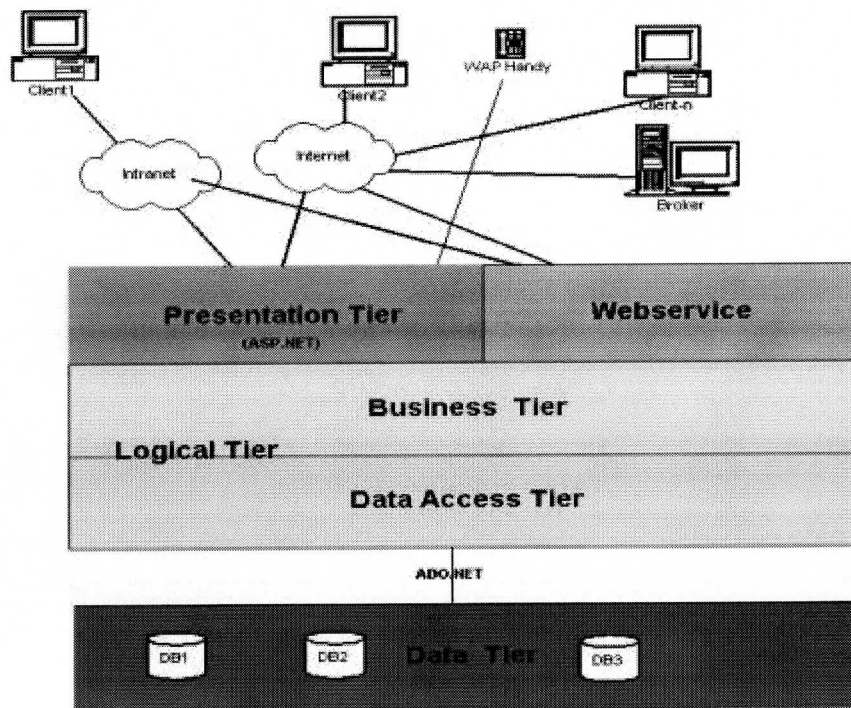
4.2. System Design.

4.2.1 Three-Tier Architecture

A 3-tier application is a program which is organized into three major disjunctive tiers. These tiers are

- Presentation Tier (Front end)
- Logical Tier (Middleware)
- Data Tier (Backend).

Each layer can be deployed in geographically separated computers in a network. Some architects divide Logic Tier in to two sub tiers Business and Data Access Tiers, in order to increase scalability and transparency. The tiers can be deployed on physically separated machines. The characteristic of the tier communication is that the tiers will communicate only to their adjacent neighbors. For an example, The Presentation Tier will interact directly with the Business Tier and not directly with Data Access or Data Tiers.



A Typical 3-Tier Architecture

Figure 7: Three tier Architecture

4.2.1.1 Data Tier

This Tier is responsible for retrieving, storing and updating from Information therefore this tier can be ideally represented through a commercial database. We consider stored procedures as a part of the Data Tier. Usage of stored procedures increases the performance and code transparency of an application

4.2.1.2 Logical Tier

This is the brain of the 3.Tier Application. Some of the advantages of discriminating logical tier with the business tier is that it Increases code transparency and it supports changes in Data Layer. You can change or alter database with out touching the Business Layer and this would be a very minimum touch up.

4.2.1.2.1 Business Tier

This sub tier contents classes to calculate aggregated values such like total revenue, cash flow and debit and this tier doesn't know about any GUI controls and how to access databases. The classes of Data Access Tier will supply the needy information from the databases to this sub tier.

4.2.1.2.2 Data Access Tier:

This tier acts as an interface to Data Tier. This tier knows how to (from which database) retrieve and store information.

4.2.1.3 Presentation Tier:

This Tier is responsible for communication with the users and web service consumers and it will use objects from Business Layer to response GUI raised events.

The system thus developed, takes into consideration, the 3 tier architecture wherein there is a mere distinction between the Presentation tier, the logical tier and the Data tier. We attempt to develop a system which provides users with a presentation tier in the form of win forms. The Logical tier resides in any of the servers of Beans'r'us which is coded using ASP .Net and so is the data tier providing data access using ADO .Net. Thus this system significantly represents a model featuring a 3 tier architecture model.

4.2.2 About Microsoft .Net Framework

The .NET Framework is the infrastructure for the new Microsoft .NET Platform. It is a common environment for building, deploying, and running Web Services and Web Applications. It contains common class libraries - like ADO.NET, ASP.NET and Windows Forms - to provide advanced standard services that can be integrated into a variety of computer systems.

The .NET Framework is language neutral. Currently it supports C++, C#, Visual Basic, JScript (The Microsoft version of JavaScript) and COBOL. Third-party languages - like Eiffel, Perl, Python, Smalltalk, and others - will also be available for building future .NET Framework applications.

The new Visual Studio.NET is a common development environment for the new .NET Framework. It provides a feature-rich application execution environment, simplified development and easy integration between a number of different development languages.

ASP .Net Overview

Active Server Pages (ASPs) are Web pages that contain server-side scripts in addition to the usual mixture of text and HTML (Hypertext Markup Language) tags. Server-side scripts are special commands you put in Web pages that are processed before the pages are sent from your Personal Web Server to the Web browser of someone who's visiting your Web site. . When you type a URL in the Address box or click a link on a Web page, you're asking a Web server on a computer somewhere to send a file to the Web browser on your computer. If that file is a normal HTML file, it looks exactly the same when your Web browser receives it as it did before the Web server sent it. After receiving the file, your Web browser displays its contents as a combination of text, images, and sounds.

In the case of an Active Server Page, the process is similar, except there's an extra processing step that takes place just before the Web server sends the file. Before the Web server sends the Active Server Page to the Web browser, it runs all server-side scripts contained in the page. Some of these scripts display the current date, time, and other information. Others process information the user has just typed into a form, such as a page in the Web site's guestbook.

To distinguish them from normal HTML pages, Active Server Pages are given the ".asp" extension.

ADO.NET Overview

ADO.NET is an evolution of the ADO data access model that directly addresses user requirements for developing scalable applications. It was designed specifically for the web with scalability, statelessness, and XML in mind.

ADO.NET uses some ADO objects, such as the Connection and Command objects, and also introduces new objects. Key new ADO.NET objects include the Dataset, Data Reader, and Data Adapter.

The important distinction between this evolved stage of ADO.NET and previous data architectures is that there exists an object -- the Dataset -- that is separate and distinct from any data stores. Because of that, the Dataset functions as a standalone entity. You can think of the Dataset as an always disconnected record set that knows nothing about the source or destination of the data it contains. Inside a Dataset, much like in a database, there are tables, columns, relationships, constraints, views, and so forth.

A Data Adapter is the object that connects to the database to fill the Dataset. Then, it connects back to the database to update the data there, based on operations performed while the Dataset held the data. In the past, data processing has been primarily connection-based. Now, in an effort to make multi-tiered apps more efficient, data processing is turning to a message-based approach that revolves around chunks of information. At the center of this approach is the DataAdapter, which provides a bridge to retrieve and save data between a DataSet and its source data store. It accomplishes this by means of requests to the appropriate SQL commands made against the data store.

The XML-based DataSet object provides a consistent programming model that works with all models of data storage: flat, relational, and hierarchical. It does this by having no 'knowledge' of the source of its data, and by representing the data that it holds as collections and data types. No matter what the source of the data within the DataSet is, it is manipulated through the same set of standard APIs exposed through the DataSet and its subordinate objects.

While the DataSet has no knowledge of the source of its data, the managed provider has detailed and specific information. The role of the managed provider is to connect, fill, and persist the DataSet to and from data stores. The OLE DB and SQL Server .NET Data Providers (System.Data.OleDb and System.Data.SqlClient) that are part of the .Net Framework provide four basic objects: the Command, Connection, DataReader and DataAdapter. In the remaining sections of this document, we'll walk through each part of the DataSet and the OLE DB/SQL Server .NET Data Providers explaining what they are, and how to program against them.

The following sections will describe some objects that have evolved, these objects are:

- Connections : For connection to and managing transactions against a database.
- Commands : For issuing SQL commands against a database.
- DataReaders : For reading a forward-only stream of data records from a SQL Server data source.
- DataSets : For storing, remoting and programming against flat data, XML data and relational data.
- DataAdapters : For pushing data into a DataSet, and reconciling data against a database.

4.2.3 Why .Net?

The .Net framework allows different programming languages and libraries to work together especially to develop applications based on windows that are easier to manage and can be connected to a network. It also creates a language independent execution environment; the .Net framework connects people, systems and devices through standard internet protocols. It also provides option to the developers to choose programming language of their interest because it supports many languages; it also supports different computing platforms.

Thus considering all the above facts, we can say that .Net is loaded with all the tools and libraries to achieve our requirements in more sophisticated and simplified way.

4.3 Database Design

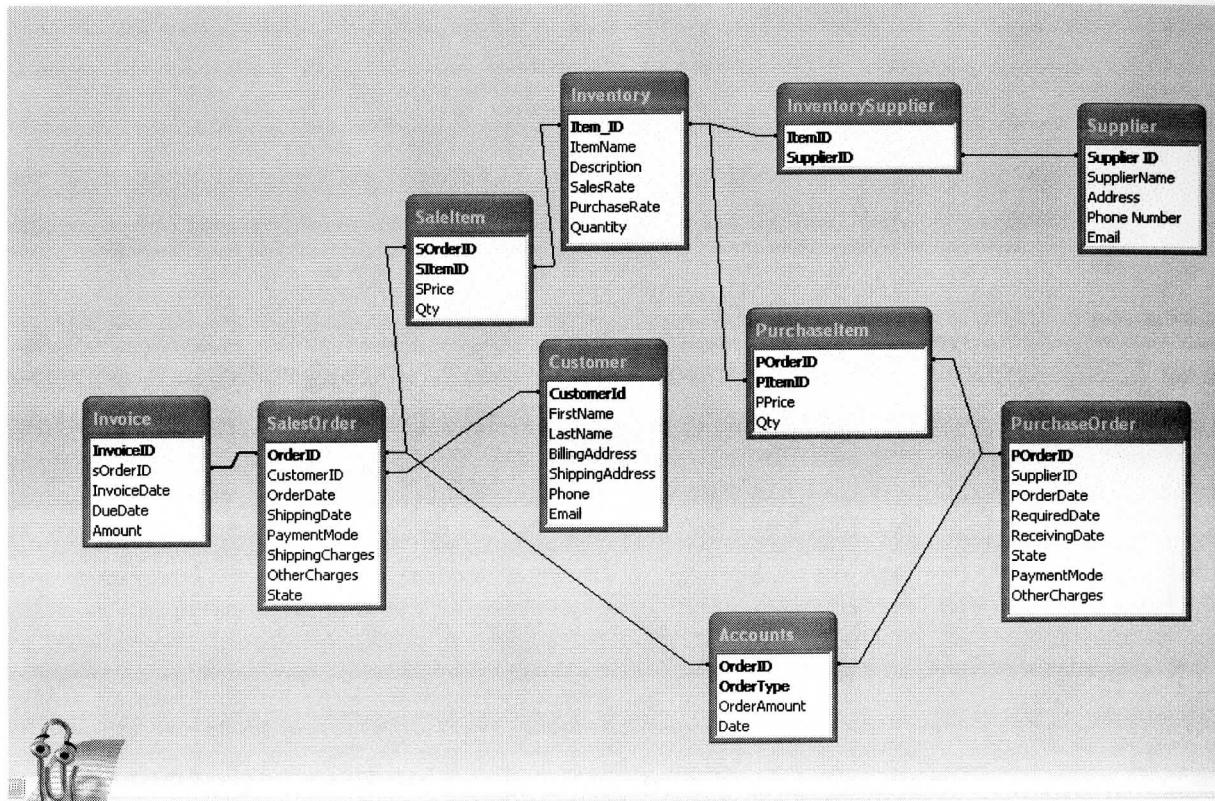


Figure 8: Data base Design

The system is supported by the Access Database as its backend or the data tier. Our database contains the following table, each of which is explained further.

i. SALEORDER

This table maintains the information of the order along with the identification details such as the OrderID and CustomerID. The following table illustrates various fields this table contains along with their description.

S.No	Field Name	Datatype	Description
1	SOrderID	Number	Unique Identification of Order – Primary Key
2	CustomerID	Number	Unique Identification of Customer – Primary Key
3	SOrderDate	Date	Order Date
4	ShippingDate	Date	Date of Shipping
5	State	Text	Maintains information about current state of the order
6	PaymentMode	Text	Records the mode of payment by the customer
7	ShippingCharges	Number	Represents additional shipping charges incurred

ii.CUSTOMER

This table maintains the information of the customer along with identification details such as the CustomerID. The following table illustrates various fields this table contains along with their description.

S.No	Field Name	Datatype	Description
1	CustomerID	Number	Unique Identification for customer– Primary Key
2	CustomerName	Text	Stores the customer name
3	Billing Address	Text	Records address where the order is to be billed
4	ShippingAddress	Text	Records address where the order is to be shipped
5	Phone	Number	Records the phone number of the customer
6	Email	Text	Stores the email address of the customer

iii. SUPPLIER

This table maintains the information of all the suppliers along with identification details such as the CustomerID. The following table illustrates various fields this table contains along with their description.

S.No	Field Name	Datatype	Description
1	SupplierID	Number	Unique Identification for supplier– Primary Key
2	SupplierName	Text	Stores the supplier or supplying company name
3	Address	Text	Records supplier address
5	Phone	Number	Records the phone number of the supplier
6	Email	Text	Stores the email address of the supplier

iv. PURCHASEORDER

This table maintains the information about the products purchased. The following table illustrates various fields this table contains along with their description.

S.No	Field Name	Datatype	Description
1	POrderID	Number	Unique identification for purchases – Primary Key
2	SupplierID	Number	Unique Identification for supplier– Foreign Key
3	PorderDate	Date	Stores the order date
5	ReceivingDate	Date	Records the date when the goods were received
6	RequiredDate	Date	Records the date when the goods are needed
7	State	Text	Stores the status of the order
8	PaymentMode	Number	Specifies how the payment would be made
9	OtherCharges	Number	Specifies additional expenses incurred

v. INVENTORY

This table maintains the information about all the items that the company is dealing with The following table illustrates various fields this table contains along with their description.

S.No	Field Name	Datatype	Description
1	ItemID	Number	Unique identification for Items – Primary Key
2	ItemName	Text	Stores the item name
3	Description	Text	Stores textual data for the item
5	SalesRate	Number	Records the sales rate of the item
6	PurchaseRate	Number	Specifies the purchase rate of the item
7	Quantity	Number	Specifies the quantity of items in stock

vi. INVENTORYSUPPLIER

This table maintains the information about the items and their suppliers relationship that the company is dealing with. The following table illustrates various fields this table contains along with their description.

S.No	Field Name	Datatype	Description
1	ItemID	Number	Unique identification for Items – Primary Key
2	SupplierID	Number	ID of suppliers

vii. INVOICE

This table maintains the information for generating invoice. The following table illustrates various fields this table contains along with their description.

S.No	Field Name	Datatype	Description
1	InvoiceID	Number	Unique Identification for the invoice– Primary Key
2	SOrderID	Text	Stores the sales order id – Foreign Key
3	InvoiceDate	Date	Records invoice date
5	DueDate	Number	Specifies maximum allowed time for payment
6	Amount	Number	Specifies the total amount levied on the transaction

viii. ACCOUNTS

This table maintains the account related information. The following table illustrates various fields this table contains along with their description.

S.No	Field Name	Datatype	Description
1	OrderID	Number	Unique Identification for the Sales or Purchase order Foreign Key
2	OrderType	Text	Records what type of order it is either Sales or purchase
3	OrderAmount	Number	Records the total amount for the transaction
5	Date	Date	Specifies the date of the transaction

ix. SALEITEM

This table maintains one to many relationship with the SALESORDER table. This table can record any number of items towards a single customer. The following table illustrates various fields this table contains along with their description.

S.No	Field Name	Datatype	Description
1	SorderID	Number	Unique Identification for the Sales order - Foreign Key
2	SitemID	Number	Records the sales item id – Foreign Key
3	Sprice	Number	Specifies the price of every item sold
5	Quantity	Number	Specifies the quantity of every item sold

x. PURCHASEITEM

This table maintains one to many relationship with the PURCHASEORDER table. This table can record any number of items towards a single inventory order. The following table illustrates various fields this table contains along with their description.

S.No	Field Name	Data type	Description
1	PorderID	Number	Unique Identification for Purchase order - Foreign Key
2	ItemID	Number	Records the item ID – Foreign Key
3	Pprice	Number	Specifies the price of every item bought
5	Quantity	Number	Specifies the quantity of every item bought

4.4 System Design:

Class Diagrams:

There are nearly six classes and their class diagrams

1.Sales order

:In the class sales order it has the attributes such as orderID,orderDATE,shippingDATE,payment,shipping charges

Salesorder
orderID:int; orderDATE:int; shippingDATE:int; shippingCHARGES:int;

2.Check inventory

:In the class Checkinventory it has following attributes itemID,Description,purchaseRATE,salesRATE,Quantity.

Checkinventory
itemID:int; Description:string; PurchaseRATE:int; salesRATE:int; Quantity:int;

3.Orderstatus

Orderstatus
OrderID:int; Status:char;

4.Newinventory

Newinventory
itemID:int;

itemNAME:char; supplierID:int; purchaseRATE:int; salesRATE:int; Quantity:int; Description:string;
--

5.Invoice

Invoice
invoiceID:int; orderID:int; invoiceDATE:int; dueDATE:int; billDATE:int;

6.Accounts

Accounts
OrderID:int; orderTYPE:string; orderAMOUNT:int; date:int;

Use Case Diagram:

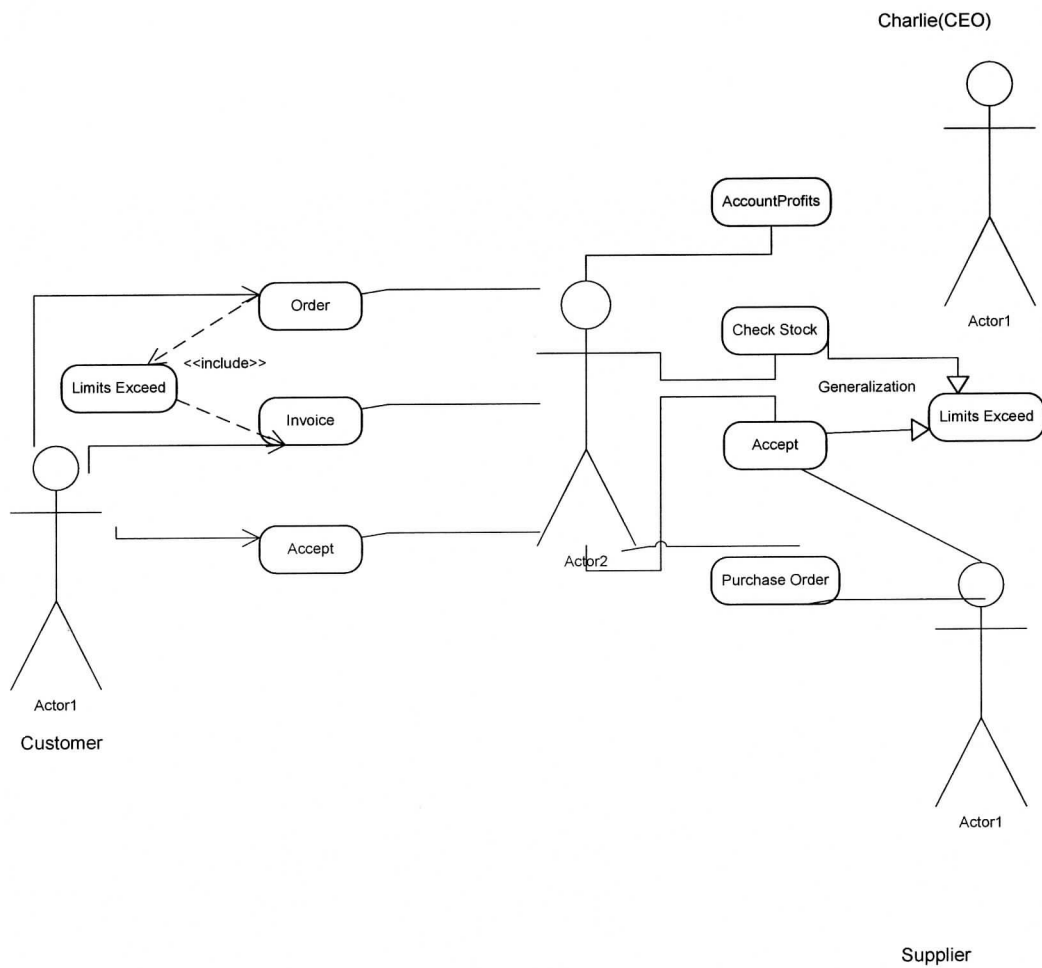


Figure 9: The explanation for use case diagram:

Normal Flow:

- 1.Customer orders and order processed by Terry
- 2.Check stock and order supplier and receives accept from supplier
- 3.Sends invoice to customer
- 4.If customer accepts goods are delivered
- 5.Rocky checks profits from accounts maintained by Terry

Alternate Flow:

- 1.If ordered items are not available in stock then orders to supplier
- 2.If supplier hadn't send goods then just waits until supplier sends
- 3.Terry sends invoice to customer
- 4.If customer hadn't accepted order is cancelled

5. IMPLEMENTING THE SYSTEM

After analyzing and designing the system. We will now bring our idea into actual implementation. While implementing the solution, we significantly focus towards fulfilling the following requirements:

- i. Develop a high quality user interface
- ii. Considering the user skills
- iii. Satisfying the requirements and working of the system.
- iv. Strictly following the design
- v. Keeping the users notified of their actions through out their transactions
- vi. Validations

5.1 At the customer's side

i. Develop a high quality user interface

User interface plays a very significant role in the world of software development. In this system, every minor aspect for developing a user friendly interface has been taken into consideration. To achieve this, various tools from Visual studio .Net has been utilized effectively, such as textboxes, combo boxes, checkboxes, radio buttons etc. The controls have been aligned in such a way so as to ensure an easy understandability. Similar controls storing similar type of data has been grouped together. Navigation to various links has been made easier with hyperlinks. Each product is presented with all its information such as its description, price & units and shipping time. This makes it easier for the customer to make decisions. Every interface is provided with short note about what is expected to be done from the customer side.

ii. Considering the user skills

After the designing of the interface, user skills are taken into consideration. It is obvious, that users need an easy way of forwarding their orders and following up, thus making the process simpler. Keeping all of these in mind, the system thus developed, has simple form fields for entering data. Data entry has been aided with combo boxes and check boxes where ever necessary. Invalid entries would present the users with error notifications, thus maintaining

consistency. Less number of fields which are of much significance is placed to avoid work load both for the customers and for the management.

iii. Satisfying the requirements and working of the system.

The next step towards implementation is to take into consideration, the requirements gathered during the analysis of the system. Having developed a good user interface, the working of all the modules is to be brought into focus.

Microsoft's Visual studio .Net provides various web and HTML tools, to develop a high level web application along with the usage of its powerful weapon, Active Server Pages (ASP .Net). Presenting online customers with simple and effective web forms has been achieved using these tools. Navigation between web pages is made possible through Hyperlinks. Transactional data is transferred and stored on the database servers through the use of ADO .Net.

After having logged on to the company's site, the customer tracks through the products. The selection of a product is made just by checking the box beside the product name. Quantity needed should be specified in the box allotted. Thus the selection of products and their quantities is rather effortless, and can be done even without much knowledge of the process. Once this is done, these values are recorded into the corresponding database and a unique identification in the form of Order ID is generated temporarily. The web form navigates to the next window, which prompts for the user identification details such as Name, Address etc in case of New Users or just the customer Id, in case of an existing user. This window also shows the products and their quantities that were selected previously. Having entered the required fields, submission of this form is done and the details are thus committed to the database. The customer is forwarded with an acknowledgement with his Order ID, for future correspondence.

Coding has been done in ASP .Net and ADO .Net to allow web logic and database connection respectively.

iv. Following the Design

The fourth module considered during the implementation, is to strictly abide by the design rules. This includes both, database and the system design. Accurate data storage, avoiding concurrent access to the database, maintaining atomicity and integrity has been the headlines. Identifying relationships between the tables of the database and storing data respectively has to be achieved through coding, which is done using the data adapters of ADO .Net.

v. Validation

Every action that the user performs should be validated so as to ensure consistency. To implement this module, we perform checks on the fields entered by the user. For example, an email address is validated by checking that it starts with a character including an underscore or a hyphen. Then it is to be concatenated with an @ sign followed by one or more characters. Finally it should terminate by a period followed by one or more characters. Thus validation proves to be an essential part of programming web applications, as they decrease the client server interactions, thereby making the process much faster. Secondly, through validation, the people sitting at the other side of the system, can ensure the accuracy of the incoming data.

5.2 Implementing the solution at the Management's side

i. High quality user interface

Every effort is taken to provide the user's at the Management side with simple and well understandable user interface. Controls provided by Win forms such as textboxes, buttons, combo boxes, radio's, data grids etc are placed in a well defined manner so as to make the presentation effective.

ii. Considering User skills

It is very much necessary that a system is developed keeping in mind the user capability. Terry – the Order Processor of Beans'r'us says that she likes to work more often with the keyboard than with the mouse. Hence taking this into consideration, we need a system which can perform most of its operations using keyboard shortcuts. All the forms that are used for entering orders, retrieving order information etc opens in a parent MDI (Multiple Document Interface) window using shortcuts. These keyboard shortcuts allow Terry to switch over to any form at any given time, perhaps making the process faster. The shortcuts are implemented using System.Windows.Forms.Shortcut property. A well defined navigation, both within forms and between the forms exists so as to enable faster processing. Checking the inventory levels, order status is made a matter of seconds, just by entering the item Id or the order Id respectively. This process is taken care by the methods provided by the ADO .Net, which is discussed in detail in the next section.

iii. Working of the system.

The system at the management side works in the following way:

- User needs to logon with specific username and password.
- She is presented with a main window, with menus. All the form windows open within the parent window.
- These windows can be accessed either by the menu control or by keyboard shortcuts.

The system at the management side consists of the following forms.

The following illustrates the implementation and working of each window:

- NewOrder
- Inventory Management
 - CheckInventory
 - NewInventory
- OrderStatus
- OrderReview
- Invoice
- Accounts

New Order

This form allows user to enter order related information such as products, order and shipping dates, customer information etc. This form is basically used by the order processor to take in orders over phones or through sales forms. Data thus entered is inserted into the **salesorder form** in the database. Tools provided by ADO .Net such as OleDb Data Adapter allows to establish a connection with the database. This adapter is also responsible for generating datasets, which contains one or more data tables for updating and retrieving data. Initially a connection is established so as to talk to the database. The code for that looks as follows:

```
string conn=@"Jet OLEDB:Global Partial Bulk Ops=2;Jet OLEDB:Registry Path=;Jet  
OLEDB:Database Locking Mode=1;Data Source=C:\Beans'r'us\Database\Beans'r'us.mdb;Jet  
OLEDB:Engine Type=5;Provider=Microsoft.Jet.OLEDB.4.0;Jet OLEDB:System database=;Jet  
OLEDB:SFP=False;persist security info=False;Extended Properties=;Mode=Share Deny  
None;Jet OLEDB:Encrypt Database=False;Jet OLEDB>Create System Database=False;Jet  
OLEDB:Don't Copy Locale on Compact=False;Jet OLEDB:Compact Without Replica  
Repair=False;User ID=Admin;Jet OLEDB:Global Bulk Transactions=1";
```

```
OleDbConnection connection=new OleDbConnection(conn);
```

Once the connection is established, datasets are generated which record the data tables on which they act on. Finally we forward the database with queries in order to perform our operations. In this case, we pass on with data insertion queries which are executed using the OleDbCommand class. The code looks something like the below:

```
OleDbCommand ol=new OleDbCommand();
    ol.Connection=connection;
    ol.CommandType=CommandType.Text;
    ol.CommandText=sql1;
    ol.CommandText=sql2;

    try
    {
        connection.Open();
        ol.ExecuteNonQuery();
        connection.Close();
    }
    catch(Exception ex1)
    {
    }
```

Apart from this, the salesorder table also contains a field called State, which shows a text concerning the status of order. This field has its own significance when the users checks in for Order Status. This field acts as follows:

- When the customer places an order, this field is being filled up with a textual description as “Order Placed”(from the neworder form)
- On generation of invoice, this field is appended with a text “Invoice Generated”. Thus for a specific entry this field will read “Order Placed Invoice Generated”.(from the invoice form)
- On payment this field is updated to “Payment Done – ready to be shipped”. While the other fields of the table, no doubt specifies when and where the order is to be shipped.

Inventory Management

Check Inventory

This form allows users to check the inventory levels. Users type in the item name for which to traverse the database. A database query is passed on to the database which returns the user with the specific record. A datagrid is used for displaying the data in the form of rows and columns. This is accomplished by the following code:

```
private void btnShow_Click(object sender, EventArgs e)
{
```



```

oleDbDataAdapter1.SelectCommand.Parameters["ItemName"].Value=txtIName.
Text;
    DSCheckInventory.Clear();
    oleDbDataAdapter1.Fill(DSCheckInventory);
}

```

In this case the query passed into the database, accepts one parameter which is the item name. The value for his itemname is passed on dynamically through the above code. Thus only specific data record is extracted and displayed.

New Inventory

The new inventory form allows user to enter or update the inventory levels. It accepts values corresponding to the item already existing in the database and also new items. A combo box provides assistance to select the items already existing in the database. On selection, all the information corresponding to the item is generated by the system itself. The user needs to enter other information like the supplier from where the item is purchased and the shipping time. Thus the purchase order details are being stored in the database, for which an invoice is generated and sent to the respective supplier. The same process is carried out if it is a new item.

OrderStatus

This form keeps track of the status of orders that have been received. The user just needs to enter the OrderID and the system returns with the whole record describing the order details and the status. This field is being updated automatically whenever a change in state occurs. OleDbDataAdapter provides a connection with the database. This adapter is loaded with a query which accepts a parameter to search the database for it. The parameter value is passed on dynamically through the text field of the form. The code looks similar to the following:

```

oleDbDataAdapter1.SelectCommand.Parameters["OrderID"].Value=txtOrderID.
Text;
    DSOrderStatus.Clear();
    oleDbDataAdapter1.Fill(DSOrderStatus);

```

OrderReview

This form presents just a photo copy of the salesorder table which present with rows ordered by orderdate. This allows the users to check if their has been any new orders, which has been placed on date. The user notes the order id and thus generate its invoice respectively.

Invoice

This form allows the user to Bill the customer. The user enters the OrderID, invoice date, bill amount and the due date. These values are further stored in the invoice table of the database.

Accounts

This form maintains the records of daily transactions. Each order whether it is a sales order or a purchase order is recorded with its order ID and Order Amount. These figures are referred and presented graphically to Rocky – The Company CEO.

5.3. Testing

5.3.1 Black Box Testing

The following modules of this project have been tested for its accuracy. We imply the Black box testing strategy; we consider the system as a black box wherein we are concerned mostly with the inputs and outputs of the system. Every output is being tracked with the respective input to check whether desired output is generated or not.

The following note illustrates the different modules tested and the results obtained.

i. User Authentication

The Login form that allows access to the system, has been tested with various combinations of usernames and passwords. Valid and invalid both types of data has been given to them, to break the key combinations.

Result : As expected, in every case the form responded only to the authorized combination of username and password.

ii. New Order

In the new orders form, every field is tested by both valid and invalid fields. Invalid data corresponding to a field value is entered, for eg. A text value in a numeric field, different format of date etc.

Result: The database responded effectively, to all the illegal entries made. For every invalid value the database query throws an exception which has been presented to the user using a message box.

iii. Inventory Management

In the check inventory form, the field value is checked both using valid and invalid user inputs.

Result: In every case, the database query responded for legal and valid data input whereas throws an exception for all invalid user inputs.

In the same way, we will be testing all the forms of the project, by providing them with invalid inputs, bulk data etc.

6. CONCLUSION

Working through all the modules of the project, we have got a good idea of the powerful features of .Net. We have seen how .Net is capable of solving business related problems. Its companion, visual studio .Net provides every tool so as to generate effective user interfaces and their validations. Communication with the databases and the web is made much more easier with ADO .Net and ASP .Net respectively. Business logic which plays one of the major roles in the 3 tier architecture can be effectively developed using the C# language. Thus, there is every reason to say that .Net has all the features built within itself, to solve any type of business problem effectively.

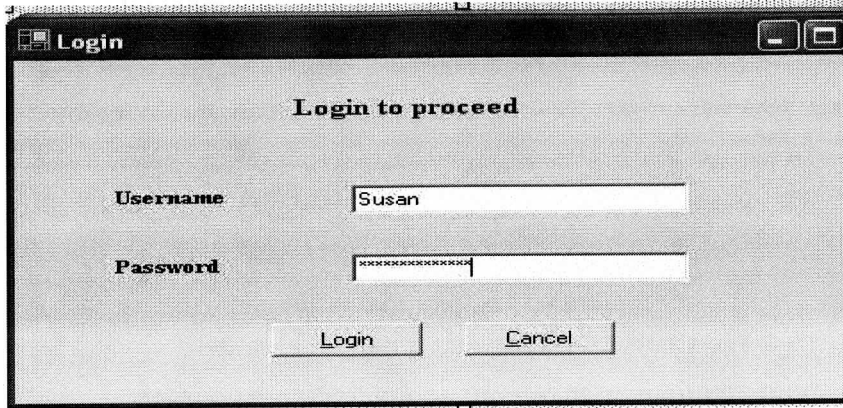
6.1 Project Constraints & Enhancements

Time plays a very significant role for generating accurate outputs. Since we are bounded by the time constraints, we restrict ourselves only to a minor part of the major idea that was thought. The system developed, though, can perform most of the operations mentioned in this report, there are still some areas where coding has to be completed. We have taken every possible effort, to bring out the best results for all the modules mentioned, but since we are not restricted to only this lecture throughout the week, we needed to divert our attentions towards other courses assignments and mini project, at times. We are still working with the functionalities of the other modules and we hope that we would bring out the best results till the time of its presentation.

This piece of work can be further enhanced by adding modules like validating credit card information online, transforming the incoming and outgoing data format using XML transformations etc.

APPENDICES:

User Interfaces



The image shows a screenshot of a Windows-style dialog box titled "Login". The dialog box has a title bar with the text "Login" and standard window control buttons (minimize, maximize, close). The main content area of the dialog box is titled "Login to proceed". Below the title, there are two input fields. The first is labeled "Username" and contains the text "Susan". The second is labeled "Password" and contains a series of asterisks "*****". At the bottom of the dialog box, there are two buttons: "Login" and "Cancel".

Figure10: Login Form

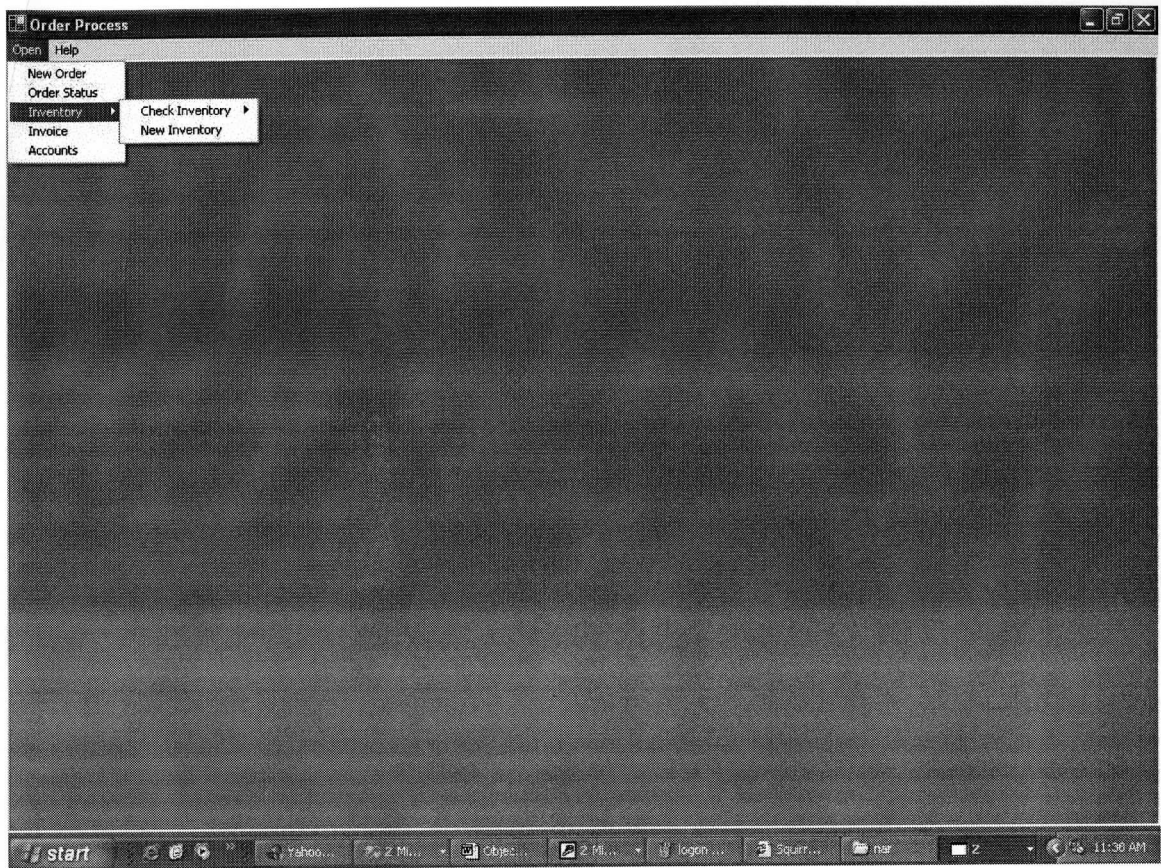


Figure11 : Order Process

Order Process - [New Orders]

Open Help

Order Details

11

Order Date Shipment Date 19-11-04

Products

Products	Qty/Kg	Products	Qty/Kg
<input type="checkbox"/> Espresso	<input type="text"/>	<input type="checkbox"/> Nescafe	<input type="text"/>
<input type="checkbox"/> Capuccino	<input type="text"/>	<input type="checkbox"/> Bru Instant	<input type="text"/>
<input type="checkbox"/> Real Tas	<input type="text"/>	<input type="checkbox"/> Black Cafe	<input type="text"/>
<input type="checkbox"/> Brazil Mix	<input type="text"/>	<input type="checkbox"/> Lipton	<input type="text"/>
<input type="checkbox"/> Amara	<input type="text"/>	<input type="checkbox"/> Zioni Instant	<input type="text"/>

Customer Details

First Name Last Name

Telephone Email

Address

Billing Address:

Street

Shipping Address:

Street

Payment Mode Shipping Charges

Figure 12: New Order Page

Order Process - [New Inventory]

Open Help

Item Details

Existing Items

Item ID

Item Name Quantity

Order Date 11/18/2004 Required Date 11/19/2004

Purchase Rate Sales Rate

Description

Supplier Information

Existing Supplier New Supplier

Supplier Detail

Address	Email	Phone Num	Supplier ID	SupplierNa
*				

Figure 13: New Inventory Page

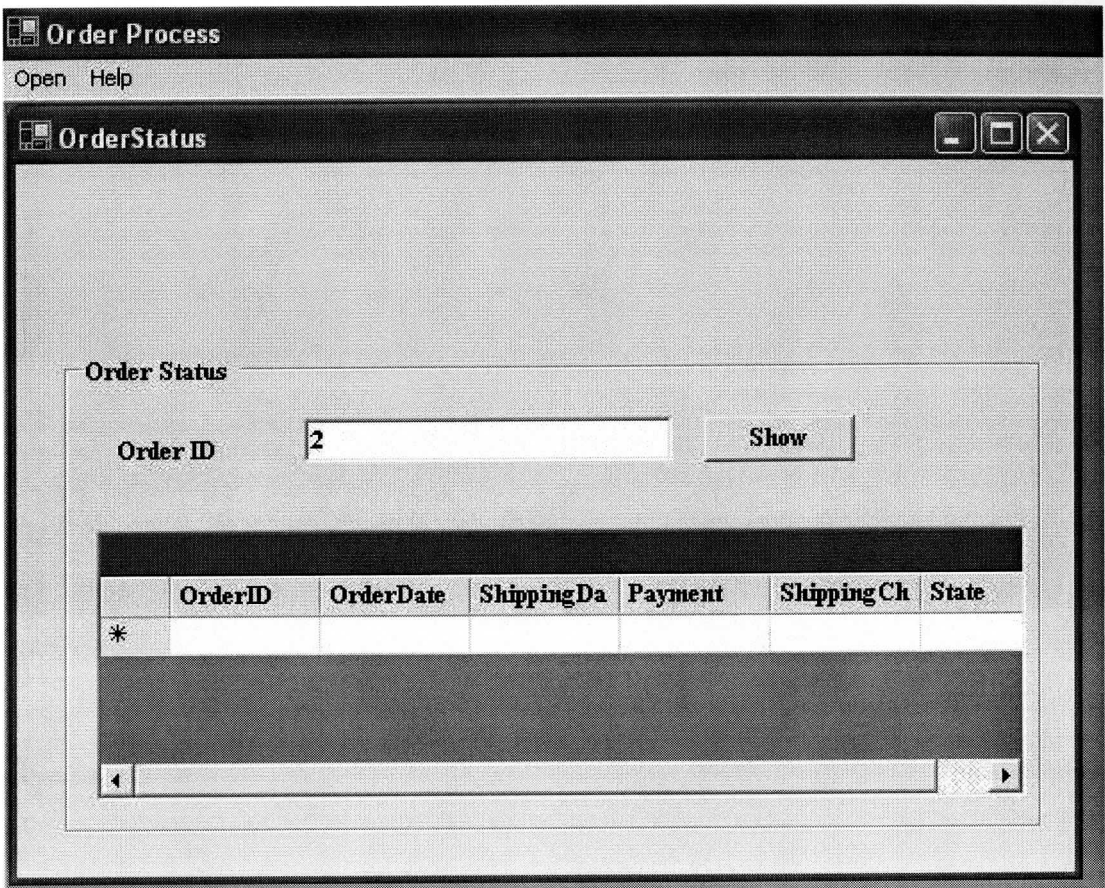


Figure 14: Order status

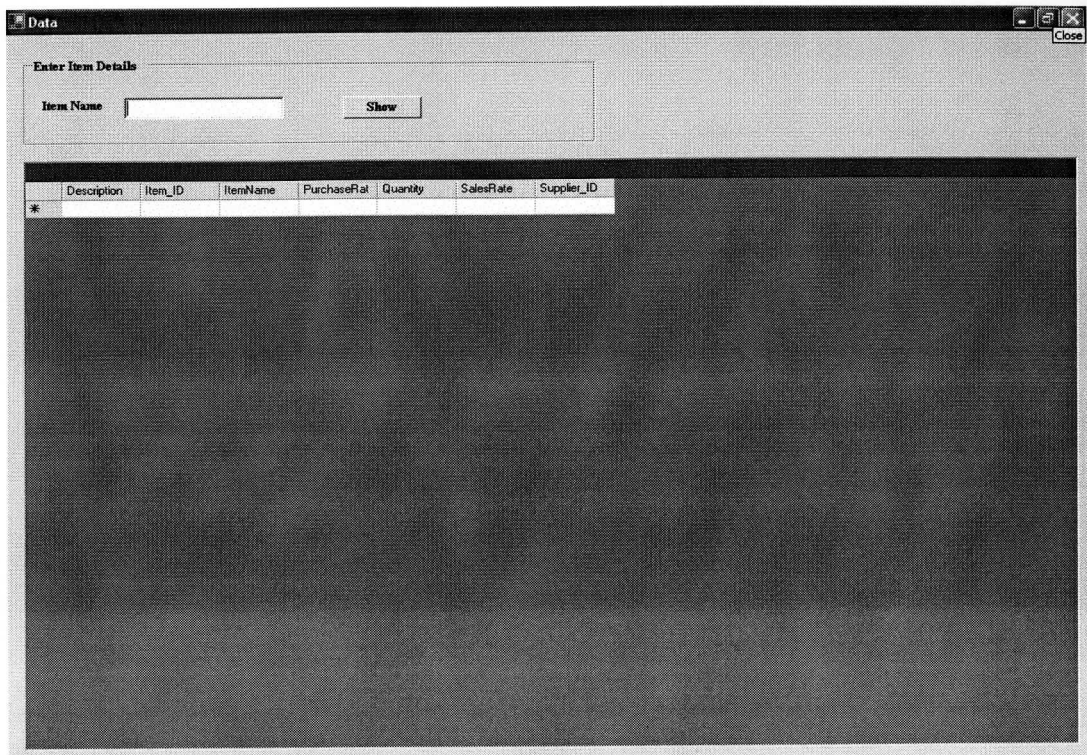


Figure15:Inventorysearchpage

Source Code

// Code for New Inventory

```
private bool ValidateInventory()
{
    string invalitems="";
    if(this.txtIID.Text=="")
        {invalitems="Item Id\n";}
    if(this.txtIName.Text=="")
        {invalitems+="Item Name\n";}
    if(this.txtIQty.Text=="")
        {invalitems+="Item Quantity\n";}
    //if(this.txtODate.Text=="")
    //    {invalitems="Item Id";}
    if(this.txtPRate.Text=="")
        {invalitems+="Purchase Rate\n";}
    //if(this.dtpRDate.Value.ToString()=="")
    //    {validated=false;}
    if(this.txtSAddress.Text=="")
        {invalitems+="Supplier Address\n";}
    if(this.txtSName.Text=="")
        {invalitems+="Supplier Name\n";}
    if(this.txtSPNo.Text=="")
        {invalitems+="Supplier Phone No.\n";}
}
```



```

        if(this.txtSRate.Text=="")
            {invaliditems+="Sale Rate\n";}
        if(invaliditems!="")
        {
            MessageBox.Show("Following Information
missing\n " + invaliditems);

            invaliditems="";
            return false;}
        else
        {return true;}
    }

    private void ShowItems()
    {
        try
        {
            OleDbCommand OleItemNameCmd=new
OleDbCommand("SELECT ItemName FROM Inventory",oleDbConnection1);
            oleDbConnection1.Open();
            //dsItemName1.Clear();
            OleDbDataReader
ItemReader=OleItemNameCmd.ExecuteReader();
            while (ItemReader.Read())
            {

                cmbItemName.Items.Add(ItemReader.GetValue(0));
                //ItemReader.NextResult();

            }
            ItemReader.Close();
        }
        catch(Exception exp)
        {
            MessageBox.Show(exp.ToString());
        }

    }

    private void NewInventory_Load(object sender,
System.EventArgs e)
    {
        ShowItems();
    }

    public static void Main()
    {
        Application.Run(new NewInventory());
    }

    private void btnRollBack_Click(object sender,
System.EventArgs e)
    {
        this.Close();
    }

```

```

private void btnCommit_Click(object sender,
System.EventArgs e)
{
    if(newval==true && ValidateInventory()==true)
    {
        string orderdate=txtODate.Text;
        string reqdate=this.dtpRqDate.Value.ToString();
        string itemname=txtIName.Text;
        string quantity=txtIQty.Text;
        string purchaserate=txtPRate.Text;
        string salesrate=txtSRate.Text;
        string desc=txtDesc.Text;
        //string sname=textBox7.Text;
        string supname=txtSName.Text;
        string phone=txtSPNo.Text;
        string email=txtSEmail.Text;
        string address=txtSAddress.Text;
        string sql="INSERT INTO
Inventory(ItemName,Description,SalesRate,PurchaseRate,Quantity)
values('"+itemname+"','"+desc+"','"+salesrate+"','"+purchaserate+"','"+
quantity+"')";
        string sql1="INSERT INTO InventorySupplier
values("+ this.txtIID.Text + "," + this.txtSID.Text + ")";
        string sql2="INSERT INTO ";
        string conn=@"Jet OLEDB:Global Partial Bulk
Ops=2;Jet OLEDB:Registry Path=;Jet OLEDB:Database Locking Mode=1;Data
Source=C:\Beans'r'us\Database\Beans'r'us.mdb;Jet OLEDB:Engine
Type=5;Provider=Microsoft.Jet.OLEDB.4.0;Jet OLEDB:System database=;Jet
OLEDB:SFP=False;persist security info=False;Extended
Properties=;Mode=Share Deny None;Jet OLEDB:Encrypt Database=False;Jet
OLEDB:Create System Database=False;Jet OLEDB:Don't Copy Locale on
Compact=False;Jet OLEDB:Compact Without Replica Repair=False;User
ID=Admin;Jet OLEDB:Global Bulk Transactions=1";
        OleDbConnection connection=new
OleDbConnection(conn);
        OleDbCommand ol=new OleDbCommand();
        ol.Connection=connection;
        ol.CommandType=CommandType.Text;
        ol.CommandText=sql;

        try
        {
            if(connection.State.ToString()=="Closed")
            {connection.Open();}
            ol.ExecuteNonQuery();
            connection.Close();
            ResetControls();
            newval=false;
            this.btnCommit.Enabled=false;
        }
        catch(Exception ex)
        {
            MessageBox.Show(ex.ToString());
        }
    }
}

```

```

    }

    private void cmbItemName_SelectedIndexChanged(object
sender, System.EventArgs e)
    {
        //MessageBox.Show(cmbItemName.SelectedItem.ToString()
);
        try{
            if(this.cmbItemName.Text!="Select Item")
            {
                OleDbCommand OleItemsCmd=new
OleDbCommand("SELECT * FROM Inventory where ItemName='" +
this.cmbItemName.Text.ToString() + "'",oleDbConnection1);
                //oleDbConnection1.Close();
                //MessageBox.Show(
oleDbConnection1.State.ToString());

                if(oleDbConnection1.State.ToString()=="Closed")
                    {oleDbConnection1.Open();}
                //dsItemName1.Clear();
                OleDbDataReader
ItemsReader=OleItemsCmd.ExecuteReader();
                //while(ItemsReader.Read())
                if(ItemsReader.FieldCount>=1)

                    {
                        ItemsReader.Read();

                        this.txtIID.Text=ItemsReader.GetValue(0).ToString();
                        this.txtIName.Text=ItemsReader.GetValue(1).ToString();
                        this.txtDesc.Text=ItemsReader.GetValue(2).ToString();
                        this.txtSRate.Text=ItemsReader.GetValue(3).ToString();
                        this.txtPRate.Text=ItemsReader.GetValue(4).ToString();
                        this.txtIQty.Text=ItemsReader.GetValue(5).ToString();

                        //cmbItemName.Items.Add(ItemsReader.GetValue(0));
                        //ItemReader.NextResult();

                        ItemsReader.Close();

                        oleDASupplier.SelectCommand.Parameters["ItemID"].Value=this.txtII
D.Text;

                        daSupplier1.Clear();
                        oleDASupplier.Fill(daSupplier1);
                    }
                this.rdbExsSupplier.Checked=true;
            }
        }
        catch(Exception exp)
        {
            MessageBox.Show(exp.ToString());
        }
    }
}

```

```

    }

    private void btnExitingItems_Click(object sender,
System.EventArgs e)
    {
        Invshow.Show();
    }

    private void rdbExsSupplier_CheckedChanged(object sender,
System.EventArgs e)
    {
        if(this.rdbExsSupplier.Checked==true)
            {this.gpbESupplier.Visible=true;
            this.gpbESupplier.Left=32;
            this.gpbESupplier.Top=472;
            this.gpbNSupplier.Visible=false;}
    }

    private void rdbNewSupplier_CheckedChanged(object sender,
System.EventArgs e)
    {
        if(this.rdbNewSupplier.Checked==true)
            {this.gpbNSupplier.Visible=true;
            this.gpbESupplier.Visible=false;
            GetSupplierID();
            }
    }

    private void dgrSupplier_Navigate(object sender,
System.Windows.Forms.NavigateEventArgs ne)
    {
    }

    private void GetSupplierID()
    {
        try
        {
            OleDbCommand OleSupIDCmd=new
OleDbCommand("SELECT MAX([Supplier ID]) FROM
Supplier",oleDbConnection1);

            if(oleDbConnection1.State.ToString()=="Closed")
                {oleDbConnection1.Open();}
            OleDbDataReader
SupIDReader=OleSupIDCmd.ExecuteReader();

            if(SupIDReader.FieldCount>=1)
            {
                SupIDReader.Read();
                int id;
                id=SupIDReader.GetInt32(0);
                id=id+1;
            }
        }
    }

```

```

        this.txtSID.Text =id.ToString();

        SupIDReader.Close();

    }
    this.rdbNewSupplier.Checked=true;
    newval=true;
}

catch(Exception exp)
{
    MessageBox.Show(exp.ToString());
}
}
private void GetItemID()
{
    try
    {

        OleDbCommand OleItemIDCmd=new
OleDbCommand("SELECT MAX([Item_ID]) FROM Inventory",oleDbConnection1);

        if(oleDbConnection1.State.ToString()=="Closed")
        {oleDbConnection1.Open();}
        OleDbDataReader
ItemIDReader=OleItemIDCmd.ExecuteReader();

        if(ItemIDReader.FieldCount>=1)

        {

            ItemIDReader.Read();
            int id;
            id=ItemIDReader.GetInt32(0);
            id=id+1;
            this.txtIID.Text =id.ToString();

            ItemIDReader.Close();

        }
        this.rdbNewSupplier.Checked=true;
        newval=true;
    }

    catch(Exception exp)
    {
        MessageBox.Show(exp.ToString());
    }
}
private void ResetControls()
{
    this.txtIID.Text="";
    this.txtIName.Text="";
    this.txtIQty.Text="";
    this.txtPRate.Text="";
    this.txtSAddress.Text="";
    this.txtDesc.Text="";
    this.txtSEmail.Text="";
}

```

```

        this.txtSID.Text="";
        this.txtSName.Text="";
        this.txtSPNo.Text="";
        this.txtSRate.Text="";
        this.cmbItemName.Text="Select Item";
    }

    private void btnNewItem_Click(object sender,
System.EventArgs e)
    {
        GetItemID();
        this.btnCommit.Enabled=true;
    }
}

```

// Code for Order Status

```

public class OrderStatus : System.Windows.Forms.Form
{
    private System.Windows.Forms.GroupBox groupBox1;
    private System.Windows.Forms.Label label1;
    private System.Data.OleDb.OleDbDataAdapter
oleDbDataAdapter1;
    private System.Data.OleDb.OleDbConnection oleDbConnection1;
    private Order_Processing.DataSet6 dataSet61;
    private System.Data.OleDb.OleDbCommand oleDbSelectCommand1;
    private System.Data.OleDb.OleDbCommand oleDbInsertCommand1;
    private System.Data.OleDb.OleDbCommand oleDbUpdateCommand1;
    private System.Data.OleDb.OleDbCommand oleDbDeleteCommand1;
    private System.Windows.Forms.Button btnOrderStatus;
    private System.Windows.Forms.DataGrid dgOrderStatus;
    private System.Windows.Forms.TextBox txtOrderID;
    /// <summary>
    /// Required designer variable.
    /// </summary>
    private System.ComponentModel.Container components = null;

    public OrderStatus()
    {
        2
        //
        // Required for Windows Form Designer support
        //
        InitializeComponent();

        //
        // TODO: Add any constructor code after
InitializeComponent call
        //
    }

    /// <summary>
    /// Clean up any resources being used.
    /// </summary>
}

```

```

protected override void Dispose( bool disposing )
{
    if( disposing )
    {
        if(components != null)
        {
            components.Dispose();
        }
    }
    base.Dispose( disposing );
}

#region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
    this.groupBox1 = new System.Windows.Forms.GroupBox();
    this.dgOrderStatus = new
System.Windows.Forms.DataGrid();
    this.dataSet61 = new Order_Processing.DataSet6();
    this.btnOrderstatus = new
System.Windows.Forms.Button();
    this.txtOrderID = new System.Windows.Forms.TextBox();
    this.labell1 = new System.Windows.Forms.Label();
    this.oleDbDataAdapter1 = new
System.Data.OleDb.OleDbDataAdapter();
    this.oleDbDeleteCommand1 = new
System.Data.OleDb.OleDbCommand();
    this.oleDbConnection1 = new
System.Data.OleDb.OleDbConnection();
    this.oleDbInsertCommand1 = new
System.Data.OleDb.OleDbCommand();
    this.oleDbSelectCommand1 = new
System.Data.OleDb.OleDbCommand();
    this.oleDbUpdateCommand1 = new
System.Data.OleDb.OleDbCommand();
    this.groupBox1.SuspendLayout();

    ((System.ComponentModel.ISupportInitialize)(this.dgOrderStatus)).
BeginInit();

    ((System.ComponentModel.ISupportInitialize)(this.dataSet61)).Begin
nInit();

    this.SuspendLayout();
    //
    // groupBox1
    //
    this.groupBox1.Controls.Add(this.dgOrderStatus);
    this.groupBox1.Controls.Add(this.btnOrderstatus);
    this.groupBox1.Controls.Add(this.txtOrderID);
    this.groupBox1.Controls.Add(this.labell1);
    this.groupBox1.Font = new System.Drawing.Font("Times
New Roman", 9.75F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));

```

```

        this.groupBox1.ForeColor =
System.Drawing.SystemColors.HotTrack;
        this.groupBox1.Location = new
System.Drawing.Point(24, 96);
        this.groupBox1.Name = "groupBox1";
        this.groupBox1.Size = new System.Drawing.Size(488,
240);

        this.groupBox1.TabIndex = 0;
        this.groupBox1.TabStop = false;
        this.groupBox1.Text = "Order Status";
        //
        // dgOrderStatus
        //
        this.dgOrderStatus.DataMember = "SalesOrder";
        this.dgOrderStatus.DataSource = this.dataSet61;
        this.dgOrderStatus.HeaderForeColor =
System.Drawing.SystemColors.ControlText;
        this.dgOrderStatus.Location = new
System.Drawing.Point(16, 88);
        this.dgOrderStatus.Name = "dgOrderStatus";
        this.dgOrderStatus.Size = new
System.Drawing.Size(464, 136);
        this.dgOrderStatus.TabIndex = 3;
        //
        // dataSet61
        //
        this.dataSet61.DataSetName = "DataSet6";
        this.dataSet61.Locale = new
System.Globalization.CultureInfo("en-US");
        //
        // btnOrderstatus
        //
        this.btnOrderstatus.Location = new
System.Drawing.Point(320, 32);
        this.btnOrderstatus.Name = "btnOrderstatus";
        this.btnOrderstatus.TabIndex = 2;
        this.btnOrderstatus.Text = "Show";
        this.btnOrderstatus.Click += new
System.EventHandler(this.button1_Click);
        //
        // txtOrderID
        //
        this.txtOrderID.Location = new
System.Drawing.Point(120, 32);
        this.txtOrderID.Name = "txtOrderID";
        this.txtOrderID.Size = new System.Drawing.Size(184,
22);

        this.txtOrderID.TabIndex = 1;
        this.txtOrderID.Text = "";
        this.txtOrderID.TextChanged += new
System.EventHandler(this.textBox1_TextChanged);
        //
        // label1
        //
        this.label1.AutoSize = true;
        this.label1.Location = new System.Drawing.Point(24,
40);

```



```

        this.label1.Name = "label1";
        this.label1.Size = new System.Drawing.Size(59, 18);
        this.label1.TabIndex = 0;
        this.label1.Text = "Order ID";
        //
        // OleDbDataAdapter1
        //
        this.OleDbDataAdapter1.DeleteCommand =
this.OleDbDeleteCommand1;
        this.OleDbDataAdapter1.InsertCommand =
this.OleDbInsertCommand1;
        this.OleDbDataAdapter1.SelectCommand =
this.OleDbSelectCommand1;
        this.OleDbDataAdapter1.TableMappings.AddRange(new
System.Data.Common.DataTableMapping[] {

            new System.Data.Common.DataTableMapping("Table",
"SalesOrder", new System.Data.Common.DataColumnMapping[] {

                new
System.Data.Common.DataColumnMapping("OrderID", "OrderID"),

                new
System.Data.Common.DataColumnMapping("OrderDate", "OrderDate"),

                new
System.Data.Common.DataColumnMapping("ShippingDate", "ShippingDate"),

                new
System.Data.Common.DataColumnMapping("Payment", "Payment"),

                new
System.Data.Common.DataColumnMapping("ShippingCharges",
"ShippingCharges"),

                new
System.Data.Common.DataColumnMapping("State", "State")
            }
        }
    );

```

```

        this.oleDbDataAdapter1.UpdateCommand =
this.oleDbUpdateCommand1;
        //
        // oleDbDeleteCommand1
        //
        this.oleDbDeleteCommand1.CommandText = @"DELETE FROM
SalesOrder WHERE (OrderID = ?) AND (OrderDate = ? OR ? IS NULL AND
OrderDate IS NULL) AND (Payment = ? OR ? IS NULL AND Payment IS NULL)
AND (ShippingCharges = ? OR ? IS NULL AND ShippingCharges IS NULL) AND
(ShippingDate = ? OR ? IS NULL AND ShippingDate IS NULL) AND (State = ?
OR ? IS NULL AND State IS NULL)";
        this.oleDbDeleteCommand1.Connection =
this.oleDbConnection1;
        this.oleDbDeleteCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Original_OrderID",
System.Data.OleDb.OleDbType.Integer, 0,
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),
((System.Byte)(0)), "OrderID", System.Data.DataRowVersion.Original,
null));
        this.oleDbDeleteCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Original_OrderDate",
System.Data.OleDb.OleDbType.DBDate, 0,
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),
((System.Byte)(0)), "OrderDate", System.Data.DataRowVersion.Original,
null));
        this.oleDbDeleteCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Original_OrderDate1",
System.Data.OleDb.OleDbType.DBDate, 0,
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),
((System.Byte)(0)), "OrderDate", System.Data.DataRowVersion.Original,
null));
        this.oleDbDeleteCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Original_Payment",
System.Data.OleDb.OleDbType.VarWChar, 50,
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),
((System.Byte)(0)), "Payment", System.Data.DataRowVersion.Original,
null));
        this.oleDbDeleteCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Original_Payment1",
System.Data.OleDb.OleDbType.VarWChar, 50,
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),
((System.Byte)(0)), "Payment", System.Data.DataRowVersion.Original,
null));
        this.oleDbDeleteCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Original_ShippingCharges",
System.Data.OleDb.OleDbType.Integer, 0,
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),
((System.Byte)(0)), "ShippingCharges",
System.Data.DataRowVersion.Original, null));
        this.oleDbDeleteCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Original_ShippingCharges1",
System.Data.OleDb.OleDbType.Integer, 0,
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),
((System.Byte)(0)), "ShippingCharges",
System.Data.DataRowVersion.Original, null));
        this.oleDbDeleteCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Original_ShippingDate",

```

```

System.Data.OleDb.OleDbType.DBDate, 0,
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),
((System.Byte)(0)), "ShippingDate",
System.Data.DataRowVersion.Original, null));
        this.oleDbDeleteCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Original_ShippingDate1",
System.Data.OleDb.OleDbType.DBDate, 0,
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),
((System.Byte)(0)), "ShippingDate",
System.Data.DataRowVersion.Original, null));
        this.oleDbDeleteCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Original_State",
System.Data.OleDb.OleDbType.VarWChar, 50,
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),
((System.Byte)(0)), "State", System.Data.DataRowVersion.Original,
null));
        this.oleDbDeleteCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Original_State1",
System.Data.OleDb.OleDbType.VarWChar, 50,
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),
((System.Byte)(0)), "State", System.Data.DataRowVersion.Original,
null));

        //
        // oleDbConnection1
        //
        this.oleDbConnection1.ConnectionString = @"Jet
OLEDB:Global Partial Bulk Ops=2;Jet OLEDB:Registry Path=;Jet
OLEDB:Database Locking Mode=1;Data
Source=""C:\Beans\r\us\Database\Beans\r\us.mdb"";Jet OLEDB:Engine
Type=5;Provider=""Microsoft.Jet.OLEDB.4.0"";Jet OLEDB:System
database=;Jet OLEDB:SFP=False;persist security info=False;Extended
Properties=;Mode=Share Deny None;Jet OLEDB:Encrypt Database=False;Jet
OLEDB>Create System Database=False;Jet OLEDB:Don't Copy Locale on
Compact=False;Jet OLEDB:Compact Without Replica Repair=False;User
ID=Admin;Jet OLEDB:Global Bulk Transactions=1";
        //
        // oleDbInsertCommand1
        //
        this.oleDbInsertCommand1.CommandText = "INSERT INTO
SalesOrder(OrderID, OrderDate, ShippingDate, Payment, ShippingCharges"
+
        ", State) VALUES (?, ?, ?, ?, ?, ?)";
        this.oleDbInsertCommand1.Connection =
this.oleDbConnection1;
        this.oleDbInsertCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("OrderID",
System.Data.OleDb.OleDbType.Integer, 0, "OrderID"));
        this.oleDbInsertCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("OrderDate",
System.Data.OleDb.OleDbType.DBDate, 0, "OrderDate"));
        this.oleDbInsertCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("ShippingDate",
System.Data.OleDb.OleDbType.DBDate, 0, "ShippingDate"));
        this.oleDbInsertCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Payment",
System.Data.OleDb.OleDbType.VarWChar, 50, "Payment"));

```

```

        this.oleDbInsertCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("ShippingCharges",
System.Data.OleDb.OleDbType.Integer, 0, "ShippingCharges"));
        this.oleDbInsertCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("State",
System.Data.OleDb.OleDbType.VarWChar, 50, "State"));
        //
        // oleDbSelectCommand1
        //
        this.oleDbSelectCommand1.CommandText = "SELECT
OrderID, OrderDate, ShippingDate, Payment, ShippingCharges, State FROM
Sal" +
        "esOrder WHERE (OrderID = ?)";
        this.oleDbSelectCommand1.Connection =
this.oleDbConnection1;
        this.oleDbSelectCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("OrderID",
System.Data.OleDb.OleDbType.Integer, 0, "OrderID"));
        //
        // oleDbUpdateCommand1
        //
        this.oleDbUpdateCommand1.CommandText = @"UPDATE
SalesOrder SET OrderID = ?, OrderDate = ?, ShippingDate = ?, Payment =
?, ShippingCharges = ?, State = ? WHERE (OrderID = ?) AND (OrderDate =
? OR ? IS NULL AND OrderDate IS NULL) AND (Payment = ? OR ? IS NULL AND
Payment IS NULL) AND (ShippingCharges = ? OR ? IS NULL AND
ShippingCharges IS NULL) AND (ShippingDate = ? OR ? IS NULL AND
ShippingDate IS NULL) AND (State = ? OR ? IS NULL AND State IS NULL)";
        this.oleDbUpdateCommand1.Connection =
this.oleDbConnection1;
        this.oleDbUpdateCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("OrderID",
System.Data.OleDb.OleDbType.Integer, 0, "OrderID"));
        this.oleDbUpdateCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("OrderDate",
System.Data.OleDb.OleDbType.DBDate, 0, "OrderDate"));
        this.oleDbUpdateCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("ShippingDate",
System.Data.OleDb.OleDbType.DBDate, 0, "ShippingDate"));
        this.oleDbUpdateCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Payment",
System.Data.OleDb.OleDbType.VarWChar, 50, "Payment"));
        this.oleDbUpdateCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("ShippingCharges",
System.Data.OleDb.OleDbType.Integer, 0, "ShippingCharges"));
        this.oleDbUpdateCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("State",
System.Data.OleDb.OleDbType.VarWChar, 50, "State"));
        this.oleDbUpdateCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Original_OrderID",
System.Data.OleDb.OleDbType.Integer, 0,
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),
((System.Byte)(0)), "OrderID", System.Data.DataRowVersion.Original,
null));
        this.oleDbUpdateCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Original_OrderDate",
System.Data.OleDb.OleDbType.DBDate, 0,

```

```
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),
((System.Byte)(0)), "OrderDate", System.Data.DataRowVersion.Original,
null));
        this.oleDbUpdateCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Original_OrderDate1",
System.Data.OleDb.OleDbType.DBDate, 0,
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),
((System.Byte)(0)), "OrderDate", System.Data.DataRowVersion.Original,
null));
        this.oleDbUpdateCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Original_Payment",
System.Data.OleDb.OleDbType.VarWChar, 50,
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),
((System.Byte)(0)), "Payment", System.Data.DataRowVersion.Original,
null));
        this.oleDbUpdateCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Original_Payment1",
System.Data.OleDb.OleDbType.VarWChar, 50,
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),
((System.Byte)(0)), "Payment", System.Data.DataRowVersion.Original,
null));
        this.oleDbUpdateCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Original_ShippingCharges",
System.Data.OleDb.OleDbType.Integer, 0,
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),
((System.Byte)(0)), "ShippingCharges",
System.Data.DataRowVersion.Original, null));
        this.oleDbUpdateCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Original_ShippingCharges1",
System.Data.OleDb.OleDbType.Integer, 0,
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),
((System.Byte)(0)), "ShippingCharges",
System.Data.DataRowVersion.Original, null));
        this.oleDbUpdateCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Original_ShippingDate",
System.Data.OleDb.OleDbType.DBDate, 0,
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),
((System.Byte)(0)), "ShippingDate",
System.Data.DataRowVersion.Original, null));
        this.oleDbUpdateCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Original_ShippingDate1",
System.Data.OleDb.OleDbType.DBDate, 0,
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),
((System.Byte)(0)), "ShippingDate",
System.Data.DataRowVersion.Original, null));
        this.oleDbUpdateCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Original_State",
System.Data.OleDb.OleDbType.VarWChar, 50,
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),
((System.Byte)(0)), "State", System.Data.DataRowVersion.Original,
null));
        this.oleDbUpdateCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Original_State1",
System.Data.OleDb.OleDbType.VarWChar, 50,
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),
((System.Byte)(0)), "State", System.Data.DataRowVersion.Original,
null));
```

```

        //
        // OrderStatus
        //
        this.AutoScaleBaseSize = new System.Drawing.Size(5,
13);
        this.ClientSize = new System.Drawing.Size(528, 358);
        this.Controls.Add(this.groupBox1);
        this.Name = "OrderStatus";
        this.Text = "OrderStatus";
        this.Load += new
System.EventHandler(this.OrderStatus_Load);
        this.groupBox1.ResumeLayout(false);

        ((System.ComponentModel.ISupportInitialize)(this.dgOrderStatus)).
EndInit();

        ((System.ComponentModel.ISupportInitialize)(this.dataSet61)).EndI
nit();
        this.ResumeLayout(false);
    }
    #endregion

    private void OrderStatus_Load(object sender,
System.EventArgs e)
    {
    }
    public static void Main()
    {
        Application.Run(new OrderStatus());
    }

    private void button1_Click(object sender, System.EventArgs
e)
    {

        OleDbDataAdapter1.SelectCommand.Parameters["OrderID"].Value=txtOr
derID.Text;
        dataSet61.Clear();
        OleDbDataAdapter1.Fill(dataSet61);

    }
    private void textBox1_TextChanged(object sender,
System.EventArgs e)
    {
    }

```

// Code for Check Inventory

```
public class CheckInventory : System.Windows.Forms.Form
```

```

    {
        private System.Windows.Forms.GroupBox groupBox1;
        private System.Windows.Forms.Label label1;
        private System.Data.OleDb.OleDbDataAdapter
oleDbDataAdapter1;
        private System.Data.OleDb.OleDbConnection oleDbConnection1;
        private Order_Processing.DataSet2 dataSet21;
        private System.Data.OleDb.OleDbCommand oleDbSelectCommand1;
        private System.Data.OleDb.OleDbCommand oleDbInsertCommand1;
        private System.Data.OleDb.OleDbCommand oleDbUpdateCommand1;
        private System.Data.OleDb.OleDbCommand oleDbDeleteCommand1;
        private System.Windows.Forms.Button btnShow;
        private System.Windows.Forms.TextBox txtIName;
        private System.Windows.Forms.DataGrid dgIDetail;
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.Container components = null;

        public CheckInventory()
        {
            //
            // Required for Windows Form Designer support
            //
            InitializeComponent();

            //
            // TODO: Add any constructor code after
InitializeComponent call
            //
        }

        public static void Main()
        {
            Application.Run(new CheckInventory());
        }
        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        protected override void Dispose( bool disposing )
        {
            if( disposing )
            {
                if(components != null)
                {
                    components.Dispose();
                }
            }
            base.Dispose( disposing );
        }

        #region Windows Form Designer generated code
        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()

```

```

    {
        this.groupBox1 = new System.Windows.Forms.GroupBox();
        this.btnShow = new System.Windows.Forms.Button();
        this.txtIName = new System.Windows.Forms.TextBox();
        this.labell1 = new System.Windows.Forms.Label();
        this.dgIDetail = new System.Windows.Forms.DataGrid();
        this.dataSet21 = new Order_Processing.DataSet2();
        this.oleDbDataAdapter1 = new
System.Data.OleDb.OleDbDataAdapter();
        this.oleDbDeleteCommand1 = new
System.Data.OleDb.OleDbCommand();
        this.oleDbConnection1 = new
System.Data.OleDb.OleDbConnection();
        this.oleDbInsertCommand1 = new
System.Data.OleDb.OleDbCommand();
        this.oleDbSelectCommand1 = new
System.Data.OleDb.OleDbCommand();
        this.oleDbUpdateCommand1 = new
System.Data.OleDb.OleDbCommand();
        this.groupBox1.SuspendLayout();

        ((System.ComponentModel.ISupportInitialize)(this.dgIDetail)).Begin
nInit();

        ((System.ComponentModel.ISupportInitialize)(this.dataSet21)).Begin
nInit();

        this.SuspendLayout();
        //
        // groupBox1
        //
        this.groupBox1.Controls.Add(this.btnShow);
        this.groupBox1.Controls.Add(this.txtIName);
        this.groupBox1.Controls.Add(this.labell1);
        this.groupBox1.Font = new System.Drawing.Font("Times
New Roman", 9.75F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
        this.groupBox1.ForeColor =
System.Drawing.SystemColors.HotTrack;
        this.groupBox1.Location = new
System.Drawing.Point(16, 24);
        this.groupBox1.Name = "groupBox1";
        this.groupBox1.Size = new System.Drawing.Size(544,
88);

        this.groupBox1.TabIndex = 0;
        this.groupBox1.TabStop = false;
        this.groupBox1.Text = "Enter Item Details";
        this.groupBox1.Enter += new
System.EventHandler(this.groupBox1_Enter);
        //
        // btnShow
        //
        this.btnShow.Location = new System.Drawing.Point(304,
40);

        this.btnShow.Name = "btnShow";
        this.btnShow.TabIndex = 2;
        this.btnShow.Text = "Show";
    }

```



```

        this.btnShow.Click += new
System.EventHandler(this.btnShow_Click);
        //
        // txtIName
        //
        this.txtIName.Location = new System.Drawing.Point(96,
40);
        this.txtIName.Name = "txtIName";
        this.txtIName.Size = new System.Drawing.Size(152,
22);
        this.txtIName.TabIndex = 1;
        this.txtIName.Text = "";
        this.txtIName.TextChanged += new
System.EventHandler(this.txtIName_TextChanged);
        //
        // label1
        //
        this.label1.AutoSize = true;
        this.label1.Location = new System.Drawing.Point(16,
40);
        this.label1.Name = "label1";
        this.label1.Size = new System.Drawing.Size(68, 18);
        this.label1.TabIndex = 0;
        this.label1.Text = "Item Name";
        //
        // dgIDetail
        //
        this.dgIDetail.Anchor =
((System.Windows.Forms.AnchorStyles)((((System.Windows.Forms.AnchorStyles.T
es.Top | System.Windows.Forms.AnchorStyles.Bottom)
| System.Windows.Forms.AnchorStyles.Left)
| System.Windows.Forms.AnchorStyles.Right)))));
        this.dgIDetail.DataMember = "";
        this.dgIDetail.DataSource = this.dataSet21.Inventory;
        this.dgIDetail.HeaderForeColor =
System.Drawing.SystemColors.ControlText;
        this.dgIDetail.Location = new
System.Drawing.Point(16, 128);
        this.dgIDetail.Name = "dgIDetail";
        this.dgIDetail.Size = new System.Drawing.Size(568,
424);
        this.dgIDetail.TabIndex = 1;
        //
        // dataSet21
        //
        this.dataSet21.DataSetName = "DataSet2";
        this.dataSet21.Locale = new
System.Globalization.CultureInfo("en-US");
        //
        // oleDbDataAdapter1
        //
        this.oleDbDataAdapter1.DeleteCommand =
this.oleDbDeleteCommand1;
        this.oleDbDataAdapter1.InsertCommand =
this.oleDbInsertCommand1;
        this.oleDbDataAdapter1.SelectCommand =
this.oleDbSelectCommand1;

```

```

        this.oleDbDataAdapter1.TableMappings.AddRange(new
System.Data.Common.DataTableMapping[] {

        new System.Data.Common.DataTableMapping("Table",
"Inventory", new System.Data.Common.DataColumnMapping[] {

                new
System.Data.Common.DataColumnMapping("Description", "Description"),

                new
System.Data.Common.DataColumnMapping("Item_ID", "Item_ID"),

                new
System.Data.Common.DataColumnMapping("ItemName", "ItemName"),

                new
System.Data.Common.DataColumnMapping("PurchaseRate", "PurchaseRate"),

                new
System.Data.Common.DataColumnMapping("Quantity", "Quantity"),

                new
System.Data.Common.DataColumnMapping("SalesRate", "SalesRate"),

                new
System.Data.Common.DataColumnMapping("Supplier_ID", "Supplier_ID"))});
        this.oleDbDataAdapter1.UpdateCommand =
this.oleDbUpdateCommand1;
        //
        // oleDbDeleteCommand1
        //
        this.oleDbDeleteCommand1.CommandText = @"DELETE FROM
Inventory WHERE (Item_ID = ?) AND (Description = ? OR ? IS NULL AND
Description IS NULL) AND (ItemName = ? OR ? IS NULL AND ItemName IS
NULL) AND (PurchaseRate = ? OR ? IS NULL AND PurchaseRate IS NULL) AND

```

```
(Quantity = ? OR ? IS NULL AND Quantity IS NULL) AND (SalesRate = ? OR  
? IS NULL AND SalesRate IS NULL) AND (Supplier_ID = ? OR ? IS NULL AND  
Supplier_ID IS NULL)";
```

```
        this.oleDbDeleteCommand1.Connection =  
this.oleDbConnection1;  
        this.oleDbDeleteCommand1.Parameters.Add(new  
System.Data.OleDb.OleDbParameter("Original_Item_ID",  
System.Data.OleDb.OleDbType.Integer, 0,  
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),  
((System.Byte)(0)), "Item_ID", System.Data.DataRowVersion.Original,  
null));  
        this.oleDbDeleteCommand1.Parameters.Add(new  
System.Data.OleDb.OleDbParameter("Original_Description",  
System.Data.OleDb.OleDbType.VarWChar, 50,  
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),  
((System.Byte)(0)), "Description", System.Data.DataRowVersion.Original,  
null));  
        this.oleDbDeleteCommand1.Parameters.Add(new  
System.Data.OleDb.OleDbParameter("Original_Description1",  
System.Data.OleDb.OleDbType.VarWChar, 50,  
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),  
((System.Byte)(0)), "Description", System.Data.DataRowVersion.Original,  
null));  
        this.oleDbDeleteCommand1.Parameters.Add(new  
System.Data.OleDb.OleDbParameter("Original_ItemName",  
System.Data.OleDb.OleDbType.VarWChar, 50,  
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),  
((System.Byte)(0)), "ItemName", System.Data.DataRowVersion.Original,  
null));  
        this.oleDbDeleteCommand1.Parameters.Add(new  
System.Data.OleDb.OleDbParameter("Original_ItemName1",  
System.Data.OleDb.OleDbType.VarWChar, 50,  
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),  
((System.Byte)(0)), "ItemName", System.Data.DataRowVersion.Original,  
null));  
        this.oleDbDeleteCommand1.Parameters.Add(new  
System.Data.OleDb.OleDbParameter("Original_PurchaseRate",  
System.Data.OleDb.OleDbType.Integer, 0,  
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),  
((System.Byte)(0)), "PurchaseRate",  
System.Data.DataRowVersion.Original, null));  
        this.oleDbDeleteCommand1.Parameters.Add(new  
System.Data.OleDb.OleDbParameter("Original_PurchaseRate1",  
System.Data.OleDb.OleDbType.Integer, 0,  
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),  
((System.Byte)(0)), "PurchaseRate",  
System.Data.DataRowVersion.Original, null));  
        this.oleDbDeleteCommand1.Parameters.Add(new  
System.Data.OleDb.OleDbParameter("Original_Quantity",  
System.Data.OleDb.OleDbType.Integer, 0,  
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),  
((System.Byte)(0)), "Quantity", System.Data.DataRowVersion.Original,  
null));  
        this.oleDbDeleteCommand1.Parameters.Add(new  
System.Data.OleDb.OleDbParameter("Original_Quantity1",  
System.Data.OleDb.OleDbType.Integer, 0,  
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),
```

```

((System.Byte)(0)), "Quantity", System.Data.DataRowVersion.Original,
null));
        this.oleDbDeleteCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Original_SalesRate",
System.Data.OleDb.OleDbType.Integer, 0,
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),
((System.Byte)(0)), "SalesRate", System.Data.DataRowVersion.Original,
null));
        this.oleDbDeleteCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Original_SalesRate1",
System.Data.OleDb.OleDbType.Integer, 0,
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),
((System.Byte)(0)), "SalesRate", System.Data.DataRowVersion.Original,
null));
        this.oleDbDeleteCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Original_Supplier_ID",
System.Data.OleDb.OleDbType.Integer, 0,
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),
((System.Byte)(0)), "Supplier_ID", System.Data.DataRowVersion.Original,
null));
        this.oleDbDeleteCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Original_Supplier_ID1",
System.Data.OleDb.OleDbType.Integer, 0,
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),
((System.Byte)(0)), "Supplier_ID", System.Data.DataRowVersion.Original,
null));
        //
        // oleDbConnection1
        //
        this.oleDbConnection1.ConnectionString = @"Jet
OLEDB:Global Partial Bulk Ops=2;Jet OLEDB:Registry Path=;Jet
OLEDB:Database Locking Mode=1;Data
Source="C:\Beans'r'us\Database\Beans'r'us.mdb";Jet OLEDB:Engine
Type=5;Provider="Microsoft.Jet.OLEDB.4.0";Jet OLEDB:System
database=;Jet OLEDB:SFP=False;persist security info=False;Extended
Properties=;Mode=Share Deny None;Jet OLEDB:Encrypt Database=False;Jet
OLEDB>Create System Database=False;Jet OLEDB:Don't Copy Locale on
Compact=False;Jet OLEDB:Compact Without Replica Repair=False;User
ID=Admin;Jet OLEDB:Global Bulk Transactions=1";
        //
        // oleDbInsertCommand1
        //
        this.oleDbInsertCommand1.CommandText = "INSERT INTO
Inventory(Description, ItemName, PurchaseRate, Quantity, SalesRate, S"
+
        "upplier_ID) VALUES (?, ?, ?, ?, ?, ?)";
        this.oleDbInsertCommand1.Connection =
this.oleDbConnection1;
        this.oleDbInsertCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Description",
System.Data.OleDb.OleDbType.VarWChar, 50, "Description"));
        this.oleDbInsertCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("ItemName",
System.Data.OleDb.OleDbType.VarWChar, 50, "ItemName"));
        this.oleDbInsertCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("PurchaseRate",
System.Data.OleDb.OleDbType.Integer, 0, "PurchaseRate"));

```

```

        this.oleDbInsertCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Quantity",
System.Data.OleDb.OleDbType.Integer, 0, "Quantity"));
        this.oleDbInsertCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("SalesRate",
System.Data.OleDb.OleDbType.Integer, 0, "SalesRate"));
        this.oleDbInsertCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Supplier_ID",
System.Data.OleDb.OleDbType.Integer, 0, "Supplier_ID"));
        //
        // oleDbSelectCommand1
        //
        this.oleDbSelectCommand1.CommandText = "SELECT
Description, Item_ID, ItemName, PurchaseRate, Quantity, SalesRate,
Supplier_ID FROM Inventory WHERE (ItemName = ?)";
        this.oleDbSelectCommand1.Connection =
this.oleDbConnection1;
        this.oleDbSelectCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("ItemName",
System.Data.OleDb.OleDbType.VarWChar, 50, "ItemName"));
        //
        // oleDbUpdateCommand1
        //
        this.oleDbUpdateCommand1.CommandText = @"UPDATE
Inventory SET Description = ?, ItemName = ?, PurchaseRate = ?, Quantity
= ?, SalesRate = ?, Supplier_ID = ? WHERE (Item_ID = ?) AND
(Description = ? OR ? IS NULL AND Description IS NULL) AND (ItemName =
? OR ? IS NULL AND ItemName IS NULL) AND (PurchaseRate = ? OR ? IS NULL
AND PurchaseRate IS NULL) AND (Quantity = ? OR ? IS NULL AND Quantity
IS NULL) AND (SalesRate = ? OR ? IS NULL AND SalesRate IS NULL) AND
(Supplier_ID = ? OR ? IS NULL AND Supplier_ID IS NULL)";
        this.oleDbUpdateCommand1.Connection =
this.oleDbConnection1;
        this.oleDbUpdateCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Description",
System.Data.OleDb.OleDbType.VarWChar, 50, "Description"));
        this.oleDbUpdateCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("ItemName",
System.Data.OleDb.OleDbType.VarWChar, 50, "ItemName"));
        this.oleDbUpdateCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("PurchaseRate",
System.Data.OleDb.OleDbType.Integer, 0, "PurchaseRate"));
        this.oleDbUpdateCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Quantity",
System.Data.OleDb.OleDbType.Integer, 0, "Quantity"));
        this.oleDbUpdateCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("SalesRate",
System.Data.OleDb.OleDbType.Integer, 0, "SalesRate"));
        this.oleDbUpdateCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Supplier_ID",
System.Data.OleDb.OleDbType.Integer, 0, "Supplier_ID"));
        this.oleDbUpdateCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Original_Item_ID",
System.Data.OleDb.OleDbType.Integer, 0,
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),
((System.Byte)(0)), "Item_ID", System.Data.DataRowVersion.Original,
null));

```

```
        this.oleDbUpdateCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Original_Description",
System.Data.OleDb.OleDbType.VarWChar, 50,
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),
((System.Byte)(0)), "Description", System.Data.DataRowVersion.Original,
null));
        this.oleDbUpdateCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Original_Description1",
System.Data.OleDb.OleDbType.VarWChar, 50,
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),
((System.Byte)(0)), "Description", System.Data.DataRowVersion.Original,
null));
        this.oleDbUpdateCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Original_ItemName",
System.Data.OleDb.OleDbType.VarWChar, 50,
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),
((System.Byte)(0)), "ItemName", System.Data.DataRowVersion.Original,
null));
        this.oleDbUpdateCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Original_ItemName1",
System.Data.OleDb.OleDbType.VarWChar, 50,
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),
((System.Byte)(0)), "ItemName", System.Data.DataRowVersion.Original,
null));
        this.oleDbUpdateCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Original_PurchaseRate",
System.Data.OleDb.OleDbType.Integer, 0,
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),
((System.Byte)(0)), "PurchaseRate",
System.Data.DataRowVersion.Original, null));
        this.oleDbUpdateCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Original_PurchaseRate1",
System.Data.OleDb.OleDbType.Integer, 0,
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),
((System.Byte)(0)), "PurchaseRate",
System.Data.DataRowVersion.Original, null));
        this.oleDbUpdateCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Original_Quantity",
System.Data.OleDb.OleDbType.Integer, 0,
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),
((System.Byte)(0)), "Quantity", System.Data.DataRowVersion.Original,
null));
        this.oleDbUpdateCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Original_Quantity1",
System.Data.OleDb.OleDbType.Integer, 0,
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),
((System.Byte)(0)), "Quantity", System.Data.DataRowVersion.Original,
null));
        this.oleDbUpdateCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Original_SalesRate",
System.Data.OleDb.OleDbType.Integer, 0,
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),
((System.Byte)(0)), "SalesRate", System.Data.DataRowVersion.Original,
null));
        this.oleDbUpdateCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Original_SalesRate1",
System.Data.OleDb.OleDbType.Integer, 0,
```

```

System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),
((System.Byte)(0)), "SalesRate", System.Data.DataRowVersion.Original,
null));
        this.oleDbUpdateCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Original_Supplier_ID",
System.Data.OleDb.OleDbType.Integer, 0,
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),
((System.Byte)(0)), "Supplier_ID", System.Data.DataRowVersion.Original,
null));
        this.oleDbUpdateCommand1.Parameters.Add(new
System.Data.OleDb.OleDbParameter("Original_Supplier_ID1",
System.Data.OleDb.OleDbType.Integer, 0,
System.Data.ParameterDirection.Input, false, ((System.Byte)(0)),
((System.Byte)(0)), "Supplier_ID", System.Data.DataRowVersion.Original,
null));
        //
        // CheckInventory
        //
        this.AutoScaleBaseSize = new System.Drawing.Size(5,
13);
        this.ClientSize = new System.Drawing.Size(592, 566);
        this.Controls.Add(this.dgIDetail);
        this.Controls.Add(this.groupBox1);
        this.Name = "CheckInventory";
        this.Text = "Data";
        this.WindowState =
System.Windows.Forms.FormWindowState.Maximized;
        this.groupBox1.ResumeLayout(false);

        ((System.ComponentModel.ISupportInitialize)(this.dgIDetail)).EndI
nit();

        ((System.ComponentModel.ISupportInitialize)(this.dataSet21)).EndI
nit();
        this.ResumeLayout(false);
    }
    #endregion

    private void txtIName_TextChanged(object sender,
System.EventArgs e)
    {
        txtIName.Focus();
    }

    private void btnShow_Click(object sender, System.EventArgs
e)
    {
        oleDbDataAdapter1.SelectCommand.Parameters["ItemName"].Value=txtI
Name.Text;
        dataSet21.Clear();
        oleDbDataAdapter1.Fill(dataSet21);
    }

```

```

        private void groupBox1_Enter(object sender,
System.EventArgs e)
        {
            }
    }
}

```

//Code of New Orders

```

private void btnOrder_Click(object sender, System.EventArgs e)
{
    string id=txtOrderID.Text;
    string odate=txtODate.Text;
    string sdate=dtpsDate.Value.ToString();
    string pay=txtPMode.Text;
    string ship=txtSCharges.Text;
    string First=txtCFName.Text;
    string Last=txtCLName.Text;
    string Phone=txtCPNo.Text;
    string Email=txtCEmail.Text;
    string Billing=txtBAddress.Text;
    string Shipping=txtSAddress.Text;

    string sql1="INSERT INTO
Customer(FirstName,LastName,BillingAddress,ShippingAddress,Phone,Email)
values('"+First+"','"+Last+"','"+Billing+"','"+Shipping+"','"+Phone+"',
'"+Email+"')";

    string sql2="INSERT INTO
SalesOrder(OrderDate,ShippingDate,Payment,ShippingCharges,State)
values('" + odate + "','"+ sdate + "','"+ pay + "','"+ ship + "','"+
+ "Order Placed" + "')" ;

    MessageBox.Show("1 Row Inserted","Operation
Succeeded");

    string conn=@"Jet OLEDB:Global Partial Bulk Ops=2;Jet
OLEDB:Registry Path=;Jet OLEDB:Database Locking Mode=1;Data
Source=C:\Beans\r'us\Database\Beans\r'us.mdb;Jet OLEDB:Engine
Type=5;Provider=Microsoft.Jet.OLEDB.4.0;Jet OLEDB:System database=;Jet
OLEDB:SFP=False;persist security info=False;Extended
Properties=;Mode=Share Deny None;Jet OLEDB:Encrypt Database=False;Jet
OLEDB>Create System Database=False;Jet OLEDB:Don't Copy Locale on
Compact=False;Jet OLEDB:Compact Without Replica Repair=False;User
ID=Admin;Jet OLEDB:Global Bulk Transactions=1";

    OleDbConnection connection=new OleDbConnection(conn);
    OleDbCommand ol=new OleDbCommand();
    ol.Connection=connection;
    ol.CommandType=CommandType.Text;
    ol.CommandText=sql1;
    ol.CommandText=sql2;
}

```



```

        try
        {
            connection.Open();
            ol.ExecuteNonQuery();
            connection.Close();
        }
        catch(Exception ex)
        {
            MessageBox.Show(ex.ToString());
        }
    }
    private void oleDbConnection1_InfoMessage(object sender,
System.Data.OleDb.OleDbInfoMessageEventArgs e)
    {
    }

    private void comboBox1_SelectedIndexChanged(object sender,
System.EventArgs e)
    {
    }

    private void oleDbDataAdapter1_RowUpdated(object sender,
System.Data.OleDb.OleDbRowUpdatedEventArgs e)
    {
    }

    private void oleDbDataAdapter1_RowUpdated_1(object sender,
System.Data.OleDb.OleDbRowUpdatedEventArgs e)
    {
    }

    private void chkLipton_CheckedChanged(object sender,
System.EventArgs e)
    {
    }

    private void dtpSDate_ValueChanged(object sender,
System.EventArgs e)
    {
    }

    private void btnNOrder_Click(object sender,
System.EventArgs e)
    {
        ShowOrderID();
        ReSetControls();
    }

```

```
private void ReSetControls()
{
    this.txtAmora.Text="";
    this.txtBAddress.Text="";
    this.txtBlackCafe.Text="";
    this.txtBrazilMix.Text="";
    this.txtBruInstant.Text="";
    this.txtCapuccino.Text="";
    this.txtCEmail.Text="";
    this.txtCFName.Text="";
    this.txtCLName.Text="";
    this.txtCPNo.Text="";
    this.txtEspresso.Text="";
    this.txtLipton.Text="";
    this.txtNescafe.Text="";

    this.txtODate.Text=System.DateTime.Today.ToShortDateString();
    this.txtPMode.Text="";
    this.txtRealTas.Text="";
    this.txtSAddress.Text="";
    this.txtSCharges.Text="";
    this.txtZioiInstant.Text="";
    this.chkAmora.Checked=false;
    this.chkBlackCafe.Checked=false;
    this.chkBrazilMix.Checked=false;
    this.chkBruInstant.Checked=false;
    this.chkCapuccino.Checked=false;
    this.chkEspresso.Checked=false;
    this.chkLipton.Checked=false;
    this.chkNescafe.Checked=false;
    this.chkRealTas.Checked=false;
    this.chkZioiInstant.Checked=false;
    this.dtpSDate.Value=System.DateTime.Today.Date;
}
}
```