

***(LIVE WEBSITE FOR A COMPANY)***  
***(S.K. ENGINEERING CO.)- AUTOMATION OF BUSINESS***

Submitted by

**BHANUJA TREHAN**  
**ISHAN JAIDKA**  
**HARPREET KAUR**  
**RAVINDER KAUR**

**Registration Number :11000153**  
**Registration Number :11000236**  
**Registration Number :11002026**  
**Registration Number: 11001113**

**Project Group Number CSE094**

**Course Code CSE445**

Under the Guidance of  
**SHILPA SHARMA MAM- Lecturer**

**School of Computer Science and Engineering**



**L** **LOVELY**  
**P** **ROFESSIONAL**  
**U** **NIVERSITY**

---

*Transforming Education Transforming India*

## **DECLARATION**

We hereby declare that the project work entitled ("Live Website for a Company :- Automation of Business ") is an authentic record of our own work carried out as requirements of Capstone Project for the award of B.Tech degree in B.TECH C.S.E from Lovely Professional University, Phagwara, under the guidance of (Ms SHILPA SHARMA Mentor), during January to April 2014. All the information furnished in this capstone project report is based on our own intensive work and is genuine.

Project Group Number: **CSE094**

Name of Student 1: Ishan Jaidka  
Registration Number: 11000236

Name of Student 2: Ravinder Kaur  
Registration Number: 11001113

Name of Student 3: Bhanuja Trehan  
Registration Number: 11000153

Name of Student 4: Harpreet Kaur  
Registration Number: 11002026

(Signature of Student 1)

Date:

(Signature of Student 2)

Date:

(Signature of Student 3)

Date:

(Signature of Student 4)

Date:

## **CERTIFICATE**

This is to certify that the declaration statement made by this group of students is correct to the best of my knowledge and belief. They have completed this Capstone Project under my guidance and supervision. The present work is the result of their original investigation, effort and study. No part of the work has ever been submitted for any other degree at any University. The Capstone Project is fit for the submission and partial fulfillment of the conditions for the award of B.Tech degree in B.TECH C.S.E from Lovely Professional University, Phagwara.

**Signature and Name of the Mentor**

**Designation**

**School of Computer Science and Engineering,**  
Lovely Professional University,  
Phagwara, Punjab.

Date : 22 Apr. 14

## **ACKNOWLEDGEMENT**

We take this opportunity to present our votes of thanks to all those guidepost who really acted as lightening pillars to enlighten our way throughout this project that has led to successful and satisfactory completion of this study.

We are really grateful to our mentor Ms. SHILPA SHARMA for providing us with all the facilities. Valuable time and advice, whole-hearted guidance, sincere cooperation and pains-taking involvement during the study and in completing the assignment of preparing the said project within time stipulated.

We are thankful to all those, particularly the various friends, who have been instrumental in creating proper, healthy and conductive environment and including new and fresh innovative ideas for us during the project, their help, it would have been extremely difficult for us to prepare the project in a time bound framework. I would also thank my institution and my faculty members without whom this project would have been a distant reality.

## **INDEX**

<b>SR.NO</b>	<b>TOPIC</b>	<b>PAGE.NO</b>
<b>1.</b>	Introduction	6-7
<b>2.</b>	Profile Of Problem	7-9
<b>3.</b>	Existing System	9-12
<b>3.1</b>	Introduction	9
<b>3.2</b>	Existing software	9
<b>3.3</b>	DFD For Present System	9-11
<b>3.4</b>	What's New In The System To Be Developed	11-12
<b>4.</b>	Problem Analysis	12-16
<b>4.1</b>	Product Defination	12-13
<b>4.2</b>	Feasibility Analysis	13-14
<b>4.3</b>	Project Plan	14-16
<b>5.</b>	Software Requirement Analysis	16-20
<b>5.1</b>	Introduction	16-17
<b>5.2</b>	General Description	17-18
<b>5.3</b>	Specific Requirements	19-20
<b>6.</b>	Design	21-67
<b>6.1</b>	System Design	21-31
<b>6.2</b>	Design Notations	32-33
<b>6.3</b>	Detailed Design	33-37
<b>6.4</b>	Flowchart	38
<b>6.5</b>	Server Side Coding	39-67
<b>7.</b>	Testing	67-71
<b>7.1</b>	Functional Testing	68
<b>7.2</b>	Structural Testing	68-69
<b>7.3</b>	Levels Of Testing	69-71
<b>8.</b>	Implementation	71-74
<b>8.1</b>	Implementation Of Project	71-72
<b>8.2</b>	Conversion Plan	72
<b>8.3</b>	Post-Implementation & Software Maintenance	73-75
<b>9.</b>	Project legacy	75
<b>9.1</b>	Current Status Of Project	75
<b>9.2</b>	Remaining Areas Of Concern	75
<b>9.3</b>	Technical & Managerial Lessons Learnt	75
<b>10.</b>	User Manual	76-79
<b>11.</b>	Source Code	79-87
<b>12.</b>	Bibliography	87

## 1. Introduction

“Live Website for a Company” is an Online or Live website of a Company Named :- “S.K. Engineering Co.” which is basically built to expand the Business of the Company S.K. Engineering Co. This website is basically to attract customers and know about this Company that they are dealing with these kind of products. Today, everyone wants that he/she should have Website so that everyone should know about him/her or about their Business. Today’s world is totally on Internet. Each and everything is becoming online. On Internet you interact with new people and get to know about them. Each and every information of a person is available on Internet today. You get to learn so many things on Internet. Today’s era is and era of Internet.

This Website basically deals with the Customers those who want to buy Lubricants that are being supplied by this Company. This Website is developed mainly Focusing on the Business aspects which simply earns by means of Clicks and Products being supplied by this Company. Whenever a user wants to buy any Product he/she needs to Login into his/her account. But if a User is New to this Site he/she needs to Register and have to create his/her profile for accessing the features of this Website i.e. he/she will be able to buy Products Online from this Company. To increase the user interactivity this Company has so many Products which the other competitors in the market do not have and web page is created very user friendly so that users should not lose their interest while interacting. They supply their Products to Retailers, Industrialists and many others. They are the Stockists too. They have already dealt with many big organizations and now they want to expand their business by using the Website so that the other organizations those who don’t know about them and the Products being supplied by them should come to know and interact with them for their business.

So looking into these factors of their Business we have built a Website for “S.K. Engineering Co.”.

You can visit this Website on the Link :-

[www.skengineeringco.in](http://www.skengineeringco.in)

It is developed with simple technological aspects which include :-

- JavaScript for HTML and CSS as designing languages.
- ASP.NET for database connectivity.
- SQL for databases.

- A Database Management (MSSQL SERVER 2008) for storing and retrieving the queries.

There are mainly three modules which are to be covered in this project and that are :-

- Admin Module for Website Management.
- User Module for Website Usage.
- Products Management.

Following are the Sub-Modules for main Modules :-

## **2. Profile Of Problem**

S.K. Engineering Co. does not have any Website that is already running Live and they wanted to Expand their Business of Lubricants/Products they deal with. This can be done through the best and easy way that is Website on Internet.

Lots of pages are integrated with the advertisements of some other Websites. Websites like Myntra.com, Jabong.com, etc. deals with the customers online by publishing their products online on their Website and customers interact with them on their Website for buying Products and the advertisements of these Websites are being integrated on Social Networking Websites like Facebook etc. In the similar way the Website of a Company S.K. Engineering Co. will deal with their Customers online by publishing their Products on “skengineeringco.in”. Well this is the starting or just an initiation, later on we will advertise it on some Social Networking Website for its popularity.

Whereas it is completely user oriented site which provides the user a true interactive medium to the S.K. Engineering Co. Company.



LOGO OF “SK ENGINEERING CO.”

## **Objective**

The objective of this project is to make a “Live Website for a Company” i.e. for S.K. Engineering Co. company. In this Website there are different types of Products that are deployed by this Company. Mainly in this project there will be 9 categories of Products and further they have Sub-categories.

So we have designed the Website basically for Products that are being deployed by the Company and for other Products in this field can also be supplied by contacting them. Even if they are not suppliers of that Product. On the whole the objective of the company is to Expand their Business as they didn't have any advertisement even. Online is the best way for advertisement as well as Expanding the Business.

## **Scope**

The scope of Website decides in which area the Website will get executed and by whom the Website will get executed. The Website will be installed on Server using IIS Server(Internet Information Services), it's a purely Microsoft's platform. The Users of this website are mainly the Existing Customers and the New Customers for buying Products that are deployed by them. The Big Companies have placed themselves in the Market with the help of Websites as their Business expansion and advertisement. The Company for which the Website has been designed is working manually for their Customers records. But now the Dynamic Website has been designed for them i.e. the records of the Customers will be saved at the Backend into the Database. And further the other industrials, retailers etc. can contact them on their Website even those who don't know about them that this Company also exists in the Market. Even they will get the lead in the Market. Actually the Website is designed according to the Company's requirement. Everyone in this modern world needs variety in almost every field and that's the reason this Website with the name “skengineeringco.in” has been designed for the company according to their needs.

## **3. EXISTING SYSTEM**

### **3.1 Introduction**



### 3.1 Introduction

The Company has decided to Expand their Business. Though this Company has Dealt with Good Companies as their Customer but now they want to further Expand their business. They basically have their business in Punjab only. And this time they want to Explore this Business of different Lubricants Supplied by them all over India and Overseas.

### 3.2 Existing Software

The company didn't have any website earlier. There was no online advertisement for the company moreover all the contact and details of the customer were managed manually. There were less chances for the company to interact with company. The customers also faced the inconvenience for the interaction earlier.

### 3.3 DFD for Website

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modelling its process aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

A DFD shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel.



Fig. 3.1 CONTEXT LEVEL DIAGRAM

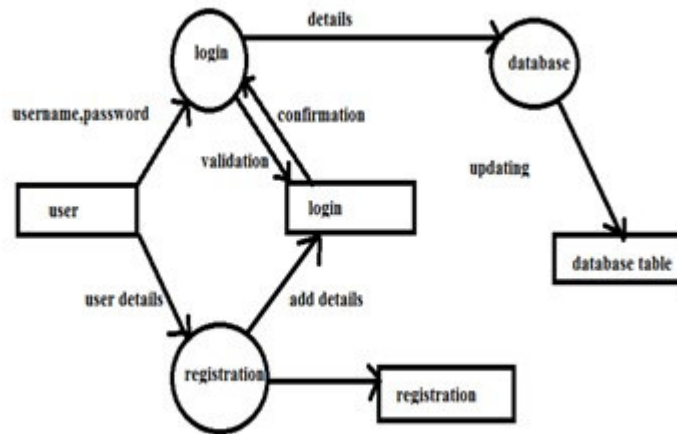


Fig. 3.2 LEVEL 1 DFD USER

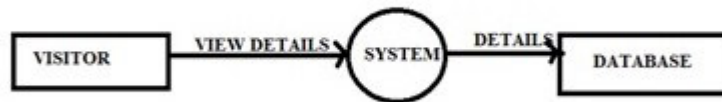


Fig. 3.3 LEVEL 1 DFD VISITOR

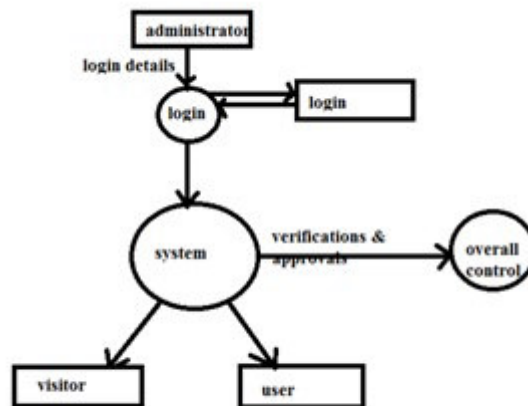


Fig. 3.4 LEVEL 1 DFD ADMINISTRATOR

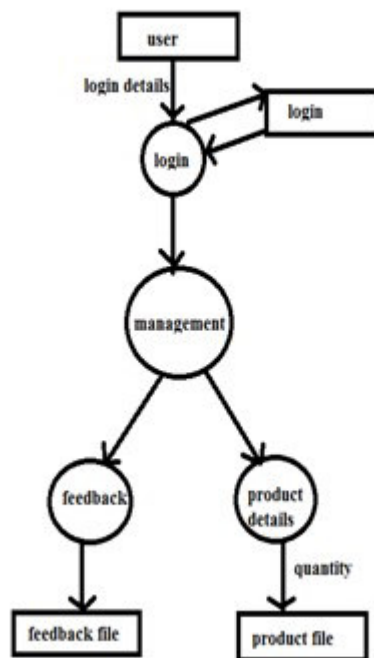


Fig. 3.5 LEVEL 2 DFD USER

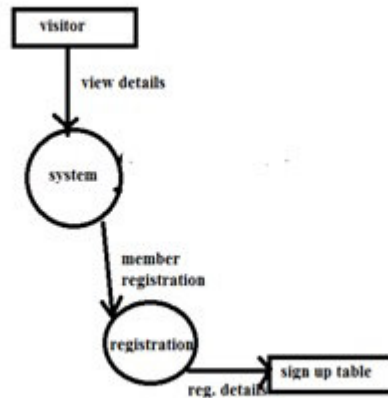


Fig.3.6 LEVEL 2 DFD VISITOR

### 3.4 What's new in the System to be developed?

This Website is designed according to the requirements of Our Client. In this Website that we have got something to study is that the Client has asked not to display the Prices of Products to the Customers as the Prices vary from Customer to Customer. They have different Customers like Industrialists, retailers, etc. Even after coming online they may get lead from

some Industrialists like they may be recommended by some Industries to other Industries and for them also they have to make different Price list of their Products.

So this is what we need to study in Future for Displaying the Prices of Products to the Customers that how we can display prices to our Customers. We have to think about it.

## **4. PROBLEM ANALYSIS**

### **4.1 Product defination**

It is a website developed using different Web Technologies together such as HTML, CSS, JavaScript and asp.net which is used to automate the process of business. It is different from other website sites in terms of efficiency, security and interactivity of the user.

#### **Efficiency**

If we talk about efficiency of site we meant by the loading time of the site. Actually what is done when a user login and clicks products user request is taken to the server and some temporary files are being downloaded in the users system in cache . If we talk about our website the efficiency factor is very high if user has limited connectivity as because of the website modules are simpler one and does not require a separate caching memory for the temporary storage of the file that's why they get loaded very easily and very fast thus preventing the users time and space. Registration is all free of cost but requires only a user's account and all the mandatory information of the user is kept safe and also the website is free of malware and viruses environment. CSS gives more control over the look and layout of Internet documents. It allows site-wide control of many elements such as font styles, page colors, lists, line spacing, margins, indents, images, etc.

#### **Security**

Security for online gaming is a very essential factor that evaluates the sites performance and growing factor because if a site is not secure than no user open it as thinking why to take the risk of letting the system to be get crashed. As a website's security level is measured in terms of free from malware and viruses which are generated during traffic patterns followed during connections during errors and bugs remained in the coding part of the software.

## **Interactivity**

If we talk about the interactivity we talk about user's interest level in the particular field.

We are going to provide a very user friendly environment so that user interacts with every corner of the site.

## **4.2 Feasibility analysis**

The main objective of feasibility study is to test the technical, operational and economical feasibility of developing a computer system. All projects are feasible, giving unlimited resources and infinite time. It is both necessary and prudent to evaluate the feasibility of the project at the system study phase itself.

The feasibility study conducted for this project involves:

- Technical feasibility
- Operational feasibility
- Economical Feasibility

Explanation is as follows:

### **Technical Feasibility:**

The considerations that are normally associated with technical feasibility include development risk, resources availability and technology. Management provides latest hardware and software facilities for the successful completion of the project.

### **Operational Feasibility:**

In the manual system, it is very difficult to maintain huge amount of pricing information of products. The development of the new system was started because of the requirements put forward by the management of the concerned department. So it is sure that the system development is operationally feasible.

This includes three major factors that are described below:-

Cost of Hardware and Software	Cost of software to be acquired to build and run the product is a onetime cost.
Benefits in reduced cost and error	Savings will be made by reduction of present system expenses, time saving and increased accuracy.

Cost avoidance	Future cost reduction in from of reduction in the number of administrative staff needed and manual records maintains in organization. Rise in cost will be avoided.

### **Economic Feasibility:**

Here the development cost is evaluated by weighing it against the ultimate benefits derived from the new system. The benefit accrued from the new system is more than the cost involved in its development as everything is related to money.

The proposed system is economically feasible because the cost involved in purchasing that hardware and the software are within approachable. The operating environment costs are marginal .The less time involved also helped in its economical feasibility.

Feasibility study is a test of system proposal according to its workability, impact on the organization, ability to meet user needs, and effective use of resources.

The objective of a feasibility study is not to solve the problem but to acquire a sense of its scope. During the study, the problem definition is crystallized and aspect of the problem to be included in the system is determined. Consequently, cost and benefits are estimated with the greater accuracy at this stage.

### **4.3 Project plan**

Project is divided into three different categories (modules) to make it the most efficient site.

**Phase 1** includes the designing part of the website which includes that how the website will look like, what menu function are there in the website, the structural overview of the website So that the website is designed to attract the user and also the interactivity of the user is also kept in mind while designing the website.

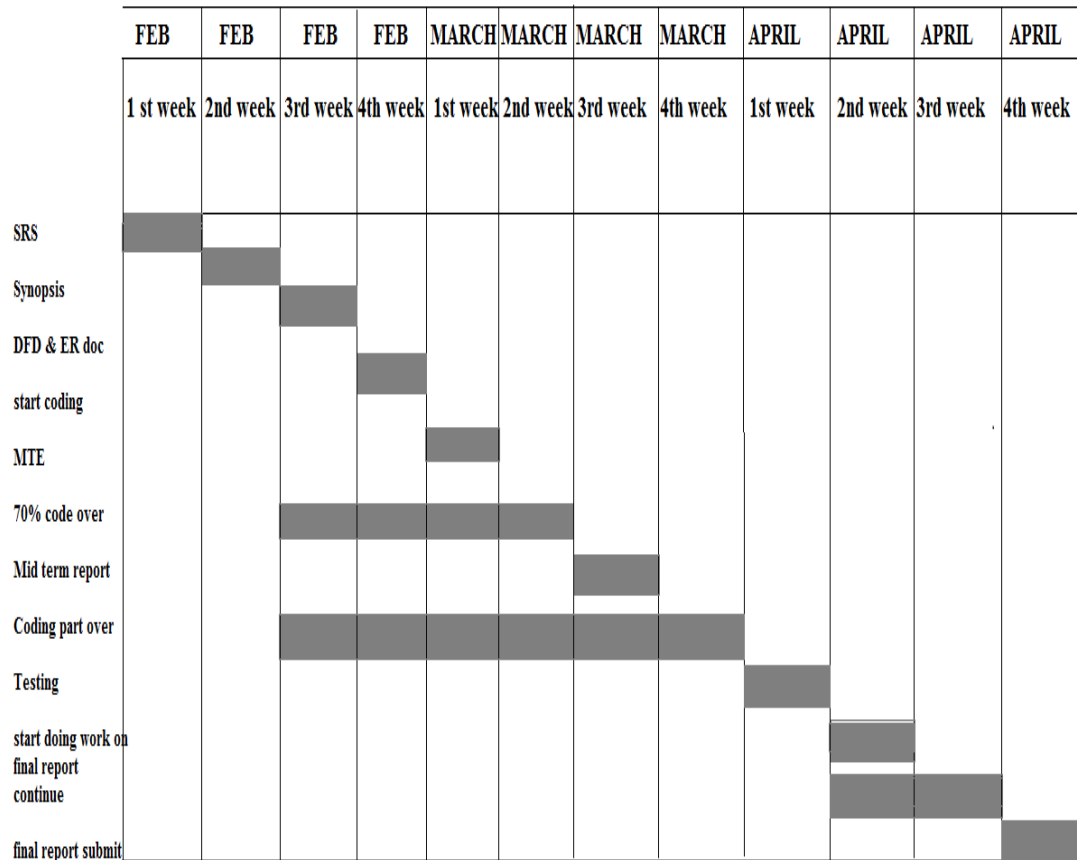
**Phase 2** includes the coding part of the website; the coding is done in a very user friendly environment so that it can support any of the web interface application like browser i.e. the coding is like that it can easily be run on any of the browser used by the user and also the coding is free of errors and bugs so that they are fully assured and free of malware and viruses that are generated during the run time phase.

**Phase 3** includes the data flow procedure during the connectivity of the user with the server while logging with a valid user id, loading of the webpage. This phase is very important in terms of accessing the database with full security user is providing his/her information which is being saved in the database so no one will have access to that information so that user's privacy option is maintained.

A Gantt chart, commonly used in project management, is one of the most popular and useful ways of showing activities (tasks or events) displayed against time. On the left of the chart is a list of the activities and along the top is a suitable time scale. Each activity is represented by a bar; the position and length of the bar reflects the start date, duration and end date of the activity. This allows you to see at a glance:

- What the various activities are
- When each activity begins and ends
- How long each activity is scheduled to last
- Where activities overlap with other activities, and by how much
- The start and end date of the whole project

## GANNT CHAT



## 5. SOFTWARE REQUIREMENT ANALYSIS

### 5.1 Introduction

This project is basically a DYNAMIC WEBSITE that is to be designed for a company who wants to expand their business online through the website. The name of the Company for which this website is going to be designed is S.K.ENGINEERING CO. This website will be linked to database and further records and orders of the customers will be managed on the database. User interface for the customers will be provided for their orders and also they can track their order. The company deals with the different LUBRICANTS.



This document is basically about how the website will be designed for the company to expand their business of different kinds of LUBRICANTS being manufactured by them. There are different categories of LUBRICANTS like hydraulic, cutting, quenching, crank case oil etc. we need different things to design a website. This website will be designed on .NET FRAMEWORK. The language that will be used is ASP.NET.

## 5.2 General Description

### Product Perspective:

Following is the context of website. Comparison b/w the Traditional system and the new system can also be cleared through the system models.



Fig. 5.2.1 Working of Website

**Functionalities:** This website will have following functionalities:-

- 1) **Administrators Management**
- 2) **Users Management**

Since this is an open source website. There are various reasons why should anyone use this website. First it's a free of cost website so that anyone can join. Second is a easy and reliable website that is very unique in its category where user can use it on any web browser and or any network interface. And third due to its open source user can take help from online assistance. The major components of the system as noted in the previous section are the website client window and the website Server window.

**(a)The Website Client** is a simple client-window (browser) form that communicates with the active website Server and its purpose is to run the game sessions that were selected and prints the results.

**(b)The Website Server** is a game and information database server that can be enabled in server mode, as long as is in that state it opens ports to accept website Client connections session statistics such as who is connected at the present time and which game is being selected for playing, saves this information in log files and load the game in the browser window.

## **Product Features**

**The major features of our website are:**

**Cross platform support:** Offers operating support for most of the known and commercial operating systems

**Language support:** Offers multiple language support for global use.

**TCP or Connectionless support:** Clients can connect to the server using a TCP connection or connectionless using a removable disk that loads the game from it.

## **Design and Implementation Constraints**

This website is created using PHP,JSP,CSS programming languages and Scripting languages with HTML and a Database server which is used for storing the files and related information's. So a minimum PC having at least 64mb of RAM and CPU over 400 MHz is required to run the website with good speed. For the connection stream TCP-IP is used as it's the common gateway for internet applications. A good internet connection is must for running the website and playing games very efficiently without any special requirements of the special hardware.

## **Assumptions and Dependencies**

For creating the website the simple HTML coding with some scripting languages is done for providing it to be supported on various browsers and games are coded in programming languages that is JavaScript. We assume that user is not interested in knowing that how they were coded at the time of their development.

## **5.3 Specific Requirements**

### **Functionality**

This section is organized by the process and features encapsulated in the website. It includes the inputs from the user, processing it accordingly and generating output to the user's window (browser).

### **Maintenance**

The data of the products and the users profile information is maintained in the database.

### **Security Considerations**

The website administrator will ensure the privacy of user profile status and information and ensure full control over its execution, so that alteration of scheduling criteria or actual resource allocation is not possible without administrator authority.

### **Software Requirements**

OPERATING SYSTEM : WINDOWS 7

BROWSER : ANY HTTP BROWSER

FRONT END : ASP.NET 4.0

DATABASE LAYER : SQL SERVER 2008

WEB SERVER : IIS

SERVER SIDE SCRIPTING : C#.NET

CLIENT SIDE SCRIPTING : JAVA SCRIPT

CONNECTION : TCP / IP

PROTOCOL : HTTP, SMTP

- 32-Bit Systems: Computer with Intel or compatible 1GHz or faster processor (2 GHz or faster is recommended. Only a single processor is supported)
- 64-Bit Systems: 1.4 GHz or higher processor (2 GHz or faster is recommended. Only a single processor is supported)
- Minimum of 512 MB of RAM (1 GB or more is recommended)
- 1 GB of free hard disk space

### Hardware Requirements

Component	Minimum	Recommended
Processor	2.5 gigahertz (GHz)	Dual processors that are each 3 GHz or faster
RAM	1 gigabyte (GB)	2 GB
Disk	NTFS file system–formatted partition with a minimum of 3 GB of free space	NTFS file system–formatted partition with 3 GB of free space plus adequate free space for website
Drive	DVD drive	DVD drive or the source copied to a local or network accessible
Display	1024 × 768	1024 × 768 or higher resolution monitor
Network	56 kilobits per second (Kbps) connection between client computers and server	56 Kbps or faster connection between client side and server

## **6. DESIGN**

### **6.1 System Design**

System design is the solution to the creation of a new system. This phase is composed of several systems. This phase focuses on the detailed implementation of the feasible system. It emphasizes on translating design specifications to performance specification.

System design has two phases of development logical and physical design.

During logical design phase the analyst describes inputs (sources), outputs (destinations), databases (data stores) and procedures (data flows) all in a format that meets the user requirements. The analyst also specifies the user needs and at a level that virtually determines the information flow into and out of the system and the data resources. Here the logical design is done through data flow diagrams and database design.

The physical design is followed by physical design or coding. Physical design produces the working system by defining the design specifications, which tell the programmers exactly what the candidate system must do. The programmers write the necessary programs that accept input from the user, perform necessary processing on accepted data through call and produce the required report on a hard copy or display it on the screen.

#### **Front End**

At the front end we have Browser as a User Interface to Interact with different Websites. Everything is clearly shown in figure below. Everything gets converted into HTML tags when we view the source of page in the Browser. What we write or create business logics in the middle ware that everything gets converted into HTML.

JavaScript, VBScript, jQuery, CSS, HTML are client-side scripts. These run at the Client side only and in the Browser. These Scripts are compiled by the Interpreter on the Browser side. Browser has an in-built Interpreter.

#### **Middle End or Middle Ware**

At the Middle Ware we use Business Logics in which we use different technologies to develop Websites like ASP.NET, PHP, RUBYONRAILS etc. These above listed are Server-side scripts, they run only on Server-side and cannot run on Client-side.

Dynamic Websites can be created with the following technologies :- JavaScript, HTML, ASP, ASP.NET, CSS, PHP, etc.

ASP.NET is a Microsoft's platform for creating Dynamic Websites. Before ASP.NET, we had ASP working at the background to develop dynamic Webpages but ASP had lot many issues with regards to Web Development.

Parallel to Visual Studio there was development of JAVA.

### **ASP.net 4.0**

ASP.NET 4.0 is not just a simple upgrade or the latest version of ASP. ASP.NET 4.0 combines unprecedented developer productivity with performance, reliability, and deployment. ASP.NET redesigns the whole process. It's still easy to grasp for new comers but it provides many new ways of managing projects. Below are the features: .

#### **Easy Programming Model**

ASP.NET 4.0 makes building real world Web applications dramatically easier. ASP.NET server controls enable an HTML-like style of declarative programming that let you build great pages with far less code than with classic ASP. Displaying data, validating user input, and uploading files are all amazingly easy. Best of all, ASP.NET pages work in all browsers including Netscape, Opera, AOL, and Internet Explorer.

#### **Flexible Language Options**

ASP.NET 4.0 lets you leverage your current programming language skills. Unlike classic ASP, which supports only interpreted VBScript and J Script, ASP.NET now supports more than 25 .NET languages (built-in support for VB.NET, C#, and JScript.NET), giving us unprecedented flexibility in the choice of language.

#### **Great Tool Support**

We can harness the full power of ASP.NET 4.0 using any text editor, even Notepad. But Visual Studio .NET adds the productivity of Visual Basic-style development to the Web. Now we can visually design ASP.NET Web Forms using familiar drag-drop-double click techniques, and enjoy full-fledged code support including statement completion and color-coding. VS.NET also provides integrated support for debugging and deploying ASP.NET

Web applications. The Enterprise versions of Visual Studio .NET deliver life-cycle features to help organizations plan, analyse, design, build, test, and coordinate teams that develop ASP.NET Web applications. These include UML class modelling, database modelling (conceptual, logical, and physical models), testing tools (functional, performance and scalability), and enterprise frameworks and templates, all available within the integrated Visual Studio .NET environment.

### **Rich Class Framework**

Application features that used to be hard to implement, or required a 3rd-party component, can now be added in just a few lines of code using the .NET Framework. The .NET Framework offers over 4500 classes that encapsulate rich functionality like XML, data access, file upload, regular expressions, image generation, performance monitoring and logging, transactions, message queuing, SMTP mail, and much more. With Improved Performance and Scalability ASP.NET4.0 lets we use serve more users with the same hardware.

### **Compiled execution**

ASP.NET 4.0 is much faster than previous versions, while preserving the "just hit save" update model of ASP. However, no explicit compile step is required. ASP.NET4.0 will automatically detect any changes, dynamically compile the files if needed, and store the compiled results to reuse for subsequent requests. Dynamic compilation ensures that the application is always up to date, and compiled execution makes it fast.

### **Rich output caching**

ASP.NET 4.0 output caching can dramatically improve the performance and scalability of the application. When output caching is enabled on a page, ASP.NET executes the page just once, and saves the result in memory in addition to sending it to the user. When another user requests the same page, ASP.NET 4.0 serves the cached result from memory without re-executing the page. Output caching is configurable, and can be used to cache individual regions or an entire page. Output caching can dramatically improve the performance of data-driven pages by eliminating the need to query the database on every request.

## **Enhanced Reliability**

ASP.NET 4.0 ensures that the application is always available to the users.

## **Memory Leak, Dead Lock and Crash Protection**

ASP.NET 4.0 automatically detects and recovers from errors like deadlocks and memory leaks to ensure our application is always available to our users. For example, say that our application has a small memory leak, and that after a week the leak has tied up a significant percentage of our server's virtual memory. ASP.NET 4.0 will detect this condition, automatically start up another copy of the ASP.NET 4.0 worker process, and direct all new requests to the new process. Once the old process has finished processing its pending requests, it is gracefully disposed and the leaked memory is released. Automatically, without administrator intervention or any interruption of service, ASP.NET 4.0 has recovered from the error.

## **Easy Deployment**

ASP.NET 4.0 takes the pain out of deploying server applications. "No touch" application deployment. ASP.NET 4.0 dramatically simplifies installation of our application. With ASP.NET, 4.0 we can deploy an entire application as easily as an HTML page, just copy it to the server. configuration settings are stored in an XML file within the application.

## **Dynamic update of running application**

ASP.NET 4.0 now lets us update compiled components without restarting the web server. In the past with classic COM components, the developer would have to restart the web server each time he deployed an update. With ASP.NET, we simply copy the component over the existing DLL, ASP.NET will automatically detect the change and start using the new code.

## **Middle End – c#.net**

In brief, C#.NET a next generation of ASP (Active Server Pages) introduced by Microsoft. Similar to previous server-side scripting technologies, C#.NET allows us to build powerful, reliable, and scalable distributed applications. C#.NET is based on the Microsoft .NET framework and uses the .NET features and tools to develop Web applications and Web services.



Even though C#.NET sounds like ASP and syntaxes are compatible with ASP but C#.NET is much more than that. It provides many features and tools, which let you develop more reliable and scalable, Web applications and Web services in less time and resources. Since C#.NET is a compiled, .NET-based environment; we can use any .NET supported languages, including VB.NET, C#, JScript.NET, and VBScript.NET to develop C#.NET applications.

## **Back End – SQL Server 2008**

At the back end we have Database. Everything gets stored into the Database. Dynamic Website is basically a Game that is totally based on Database. At the back end we have used Microsoft SQL Server 2008.

Dynamic websites normally interacts with Database at the back end and then display all the information on the Specific pages. The same concept is used on Gmail, Hotmail, Facebook, Twitter, etc.

While Website gets loaded onto the Browser everything gets converted into HTML tags and this all is done by the Browser itself. The Website we have developed is basically 3-Tier because it has 1.) User Interface, 2.) Middle Ware, 3.) Back End(Database)

Microsoft SQL Server 2008 Management Studio Express is a free, integrated environment for accessing, configuring, managing, administering, and developing all components of SQL Server, as well as combining a broad group of graphical tools and rich script editors that provide access to SQL Server to developers and administrators of all skill levels.

### **User-defined functions**

SQL 2008 Server has always provided the ability to store and execute SQL code routines via stored procedures. In addition, SQL Server has always supplied a number of built-in functions. Functions can be used almost anywhere an expression can be specified in a query. This was

one of the shortcomings of stored procedures—they couldn't be used inline in queries in select lists, where clauses, and so on. Perhaps we want to write a routine to calculate the last business day of the month. With a stored procedure, we have to exec the procedure, passing in the current month as a parameter and returning the value into an output variable, and then use the variable in our queries. If only we could write our own function that we could use directly in the query just like a system function. In SQL Server 2000, we have.

## **Indexed views**

Views are often used to simplify complex queries, and they can contain joins and aggregate functions. However, in the past, queries against views were resolved to queries against the underlying base tables, and any aggregates were recalculated each time we ran a query against the view. In SQL Server 2000 Enterprise or Developer Edition, we can define indexes on views to improve query performance against the view. When creating an index on a view, the result set of the view is stored and indexed in the database. Existing applications can take advantage of the performance improvements without needing to be modified.

## **Distributed partitioned views**

SQL Server provided the ability to create partitioned views using the UNION ALL statement in a view definition. It was limited, however, in that all the tables had to reside within the same SQL Server where the view was defined. SQL Server 2008 expands the ability to create partitioned views by allowing us to horizontally partition tables across multiple SQL Servers. The feature helps to scale out one database server to multiple database servers, while making the data appear as if it comes from a single table on a single SQL Server. In addition, partitioned views can now be updated.

### **Text in row data**

In previous versions of SQL Server, text and image data was always stored on a separate page chain from where the actual data row resided. The data row contained only a pointer to the text or image page chain, regardless of the size of the text or image data. SQL Server 2008 provides a new text in row table option that allows small text and image data values to be placed directly in the data row, instead of requiring a separate data page. This can reduce the amount of space required to store small text and image data values, as well as reduce the amount of I/O required to retrieve rows containing small text and image data values.

## **Cascading constraints**

In previous versions of SQL Server, referential integrity (RI) constraints were restrictive only. If an insert, updates, or delete operation violated referential integrity, it was aborted with an error message. SQL Server 2008 provides the ability to specify the action to take when a column referenced by a foreign key constraint is updated or deleted. We can still abort the update or delete if related foreign key records exist by specifying the NO ACTION

option, or we can specify the new CASCADE option, which will cascade the update or delete operation to the related foreign key records.

### **Multiple SQL server instances**

Previous versions of SQL Server supported the running of only a single instance of SQL Server at a time on a computer. Running multiple instances or multiple versions of SQL Server required switching back and forth between the different instances, requiring changes in the Windows registry.

SQL Server 2000 provides support for running multiple instances of SQL Server on the same system. This allows us to simultaneously run one instance of SQL Server 6.5 or 7.0 along with one or more instances of SQL Server 2008. Each SQL Server instance runs independently of the others and has its own set of system and user databases, security configuration, and so on. Applications can connect to the different instances in the same way they connect to different SQL Servers on different machines.

### **XMLsupport**

Extensible Markup Language has become a standard in Web-related programming to describe the contents of a set of data and how the data should be output or displayed on a Web page. XML, like HTML, is derived from the Standard Generalize Markup Language (SGML). When linking a Web application to SQL Server, a translation needs to take place from the result set returned from SQL Server to a format that can be understood and displayed by a Web application. Previously, this translation needed to be done in a client application.

### **Log shipping**

The Enterprise Edition of SQL Server 2008 now supports log shipping, which we can use to copy and load transaction log backups from one database to one or more databases on a constant basis. This allows you to have a primary read/write database with one or more read-only copies of the database that are kept synchronized by restoring the logs from the primary database. The destination database can be used as a warm standby for the primary database, for which we can switch users over in the event of a primary database failure. Additionally, log shipping provides a way to offload read-only query processing from the primary database to the destination database.

## **ADO.Net**

Most applications need data access at one point of time making it a crucial component when working with applications. Data access is making the application interact with a database, where all the data is stored. Different applications have different requirements for database access. ASP.NET uses ADO .NET (Active X Data Object) as its data access and manipulation protocol which also enables us to work with data on the Internet.

### **ADO.NET Data Architecture**

Data Access in ADO.NET relies on two components: Data Set and Data Provider.

#### **1. Data Set**

The dataset is a disconnected, in-memory representation of data. It can be considered as a local copy of the relevant portions of the database. The Data Set is persisted in memory and the data in it can be manipulated and updated independent of the database. When the use of this Data Set is finished, changes can be made back to the central database for updating. The data in Data Set can be loaded from any valid data source like Microsoft SQL server database, an Oracle database or from a Microsoft Access database.

#### **2. Data Provider**

The Data Provider is responsible for providing and maintaining the connection to the database. A Data Provider is a set of related components that work together to provide data in an efficient and performance driven manner. The .NET Framework currently comes with two Data Providers: the SQL Data Provider which is designed only to work with Microsoft's SQL Server 7.0 or later and the OleDb Data Provider which allows us to connect to other types of databases like Access and Oracle. Each Data Provider consists of the following component classes:

The Connection object which provides a connection to the database. The Command object which is used to execute a command. The Data Reader object which provides a forward-only, read only, connected record set. The DataAdapter object which populates a disconnected DataSet with data and performs update.

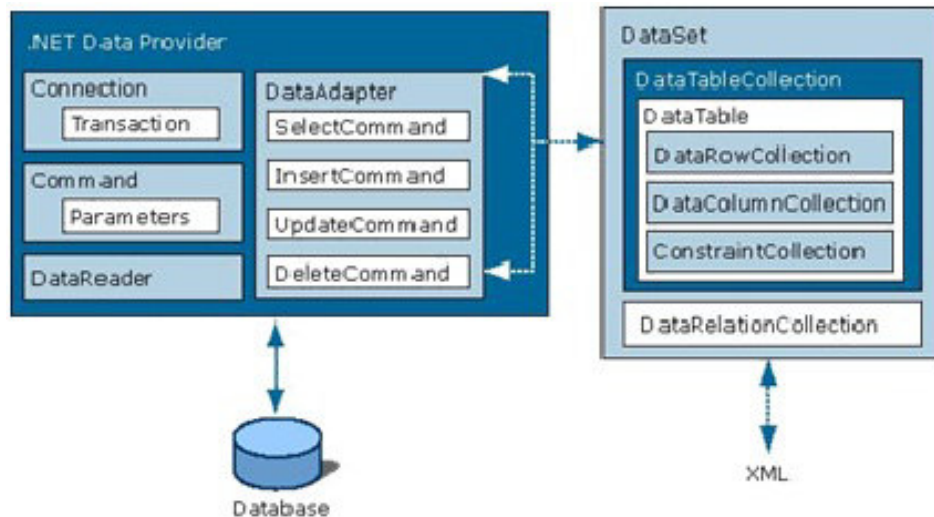


Fig. 6.1 Architectural Model

Data access with ADO.NET can be summarized as follows:

A connection object establishes the connection for the application with the database. The command object provides direct execution of the command to the database. If the command returns more than a single value, the command object returns a Data Reader to provide the data. Alternatively, the Data Adapter can be used to fill the Dataset object. The database can be updated using the command object or the Data Adapter.

Component classes that make up the Data Providers

### 1. The Connection Object

The Connection object creates the connection to the database. Microsoft Visual Studio .NET provides two types of Connection classes: the **SqlConnection** object, which is designed specifically to connect to Microsoft SQL Server 7.0 or later, and the **OleDbConnection** object, which can provide connections to a wide range of database types like Microsoft Access and Oracle. The Connection object contains all of the information required to open a connection to the database.

### 2. The Command Object

The Command object is represented by two corresponding classes: **SqlCommand** and **OleDbCommand**. Command objects are used to execute commands to a database across a data connection. The Command objects can be used to execute stored procedures on the database, SQL commands, or return complete tables directly.

### 3. The DataReader Object

The DataReader object provides a forward-only, read-only, connected stream recordset from a database. Unlike other components of the Data Provider, DataReader objects cannot be directly instantiated. Rather, the DataReader is returned as the result of the Command object's ExecuteReader method. The SqlCommand.ExecuteReader method returns a SqlDataReader object, and the OleDbCommand.ExecuteReader method returns an OleDbDataReader object. The DataReader can provide rows of data directly to application logic when we do not need to keep the data cached in memory.

### 4. The DataAdapter Object

The DataAdapter is the class at the core of ADO .NET's disconnected data access. It is essentially the middleman facilitating all communication between the database and a DataSet. The DataAdapter is used either to fill a DataTable or DataSet with data from the database with its Fill method. After the memory-resident data has been manipulated, the DataAdapter can commit the changes to the database by calling the Update method. The DataAdapter provides four properties that represent database commands:

SelectCommand, InsertCommand, DeleteCommand and UpdateCommand

When the Update method is called, changes in the DataSet are copied back to the database and the appropriate InsertCommand, DeleteCommand, or UpdateCommand is executed

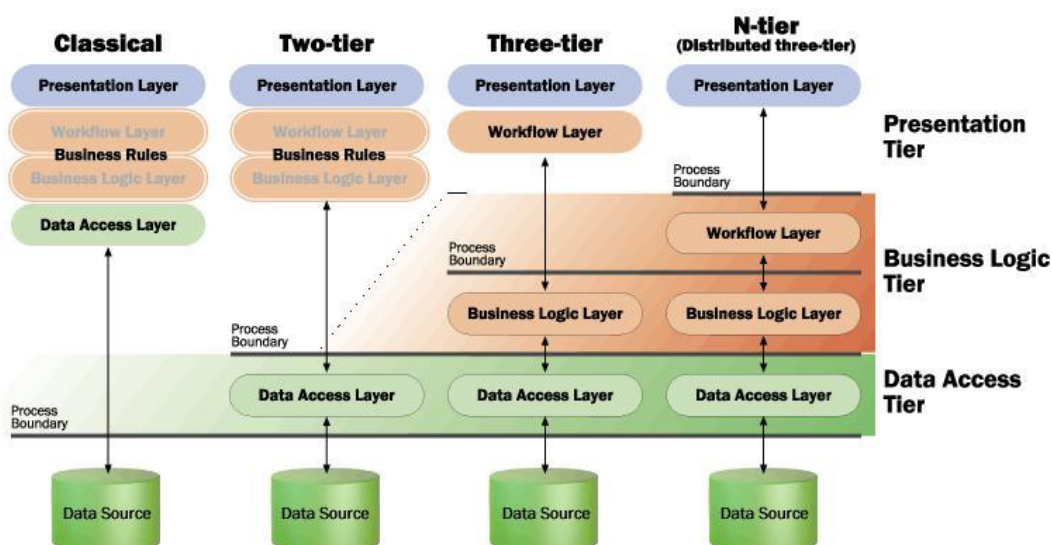


Fig. 6.2 The Architectural Model

## **Phase 1: Classic**

In the classic model, note how all layers are held within the application itself. This architecture would be very awkward to maintain in a large-scale environment unless extreme care was taken to fully encapsulate or modularize the code. Because Phase 1 of the Duwamish Books sample focuses on a small retail operation, this type of design is perfectly acceptable. It's easy to develop and, in the limited environment of a single retail outlet, easy to maintain.

In Phase 1, we deliver the basic functionality and documentation of the code and design issues.

## **Phase 2: Two-tier**

Phase 2 moves to a two-tier design, as we break out the data access code into its own layer. By breaking out this layer, we make multiple-user access to the data much easier to work with. The developer does not have to worry about record locking, or shared data, because all data access is encapsulated and controlled within the new tier.

## **Phase 3: Logical three-tier and physical three-tier**

The business rules layer contains not only rules that determine what to do with data, but also how and when to do it. For an application to become scalable, it is often necessary to split the business rules layer into two separate layers: the client-side business logic, which we call workflow, and the server-side business logic. Although we describe these layers as client and server-side, the actual physical implementations can vary. Generally, workflow rules govern user input and other processes on the client, while business logic controls the manipulation and flow of data on the server.

Phase 3 of the Duwamish Books sample breaks out the business logic into a COM[] component to create a logical three-tier application. Our second step in creating a three-tier application is to provide a physical implementation of the architecture. To distribute the application across a number of computers, we implement Microsoft Transaction Server in Phase 3.5. The application becomes easier to maintain and distribute, as a change to the business rules affects a smaller component, not the entire application. This involves some fairly lengthy analysis because the business rules in Phase 1 were deliberately not encapsulated.

## 6.2 DESIGN NOTATIONS

Entity Relationship Diagrams (ERDs) illustrate the logical structure of databases.

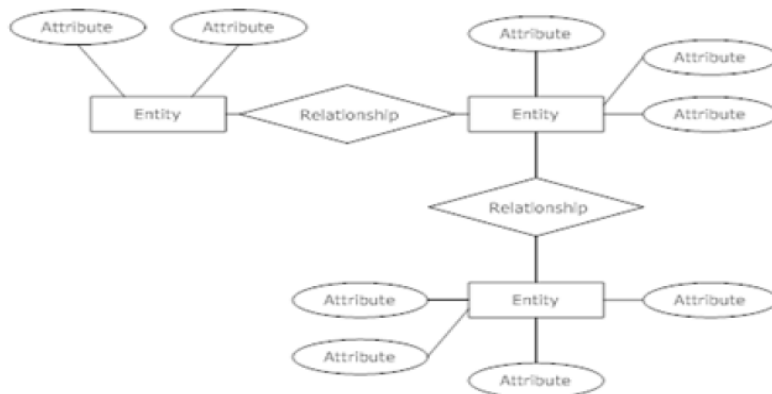


Fig. 6.2.1 ER Diagram of Database

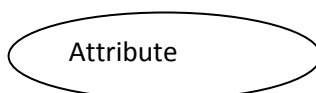
### Entity

An entity is an object or concept about which you want to store information.



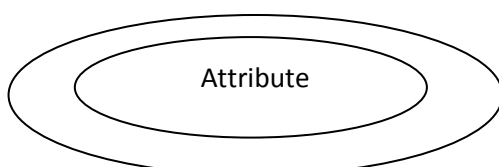
### Key attribute

A key attribute is the unique, distinguishing characteristic of the entity.



### Multi-valued attribute

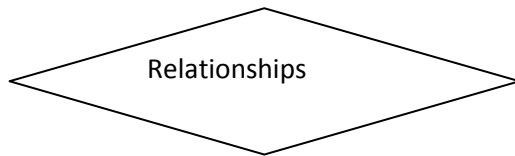
A multi-valued attribute can have more than one value.





## **Relationships**

Relationships illustrate how two entities share information in the database structure.



## **Cardinality**

Cardinality specifies how many instances of an entity relate to one instance of another entity.

## **Recursive relationship**

In some cases, entities can be self-linked.

## **6.3 DETAILED DESIGN**

### **DFD**

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modelling its process aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

A DFD shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel

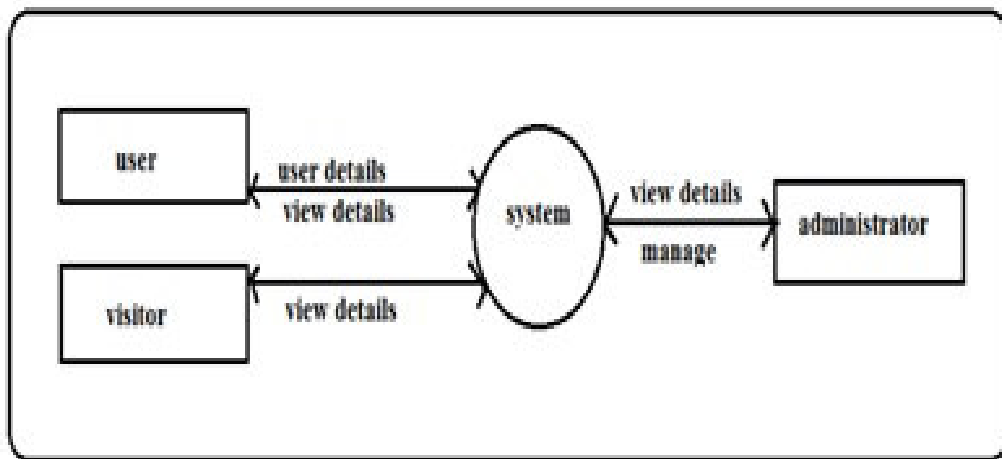


Fig. 6.3.1 Context Level Diagram

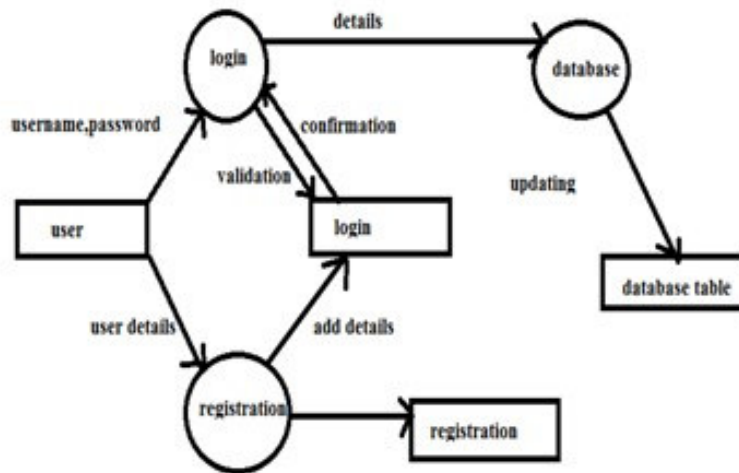


Fig. 6.3.2 LEVEL 1 DFD USER

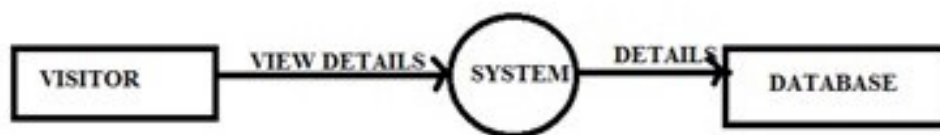


Fig. 6.3.3 LEVEL 1 DFD VISITOR

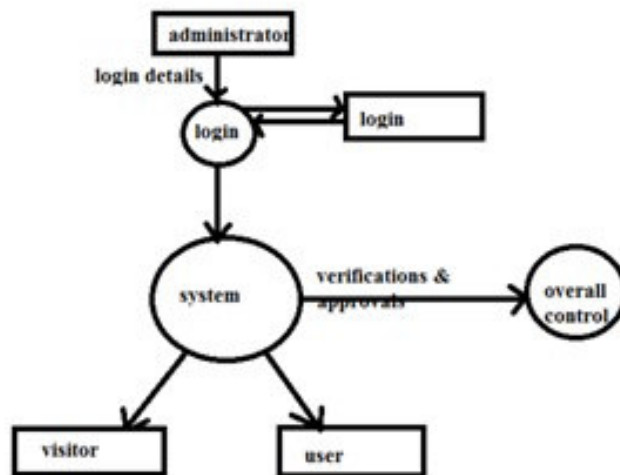


Fig. 6.3.4 LEVEL 1 DFD ADMINISTRATOR

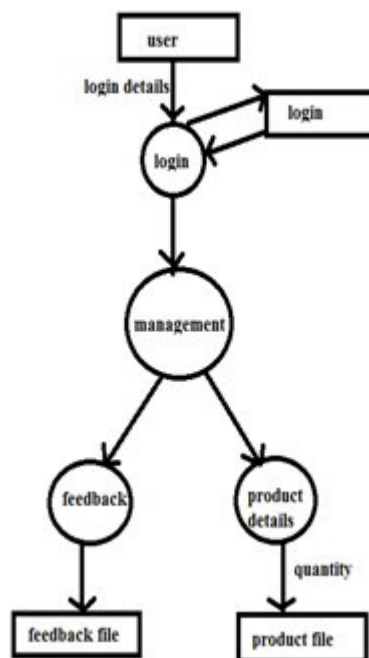


Fig. 6.3.5 LEVEL2 DFD USER

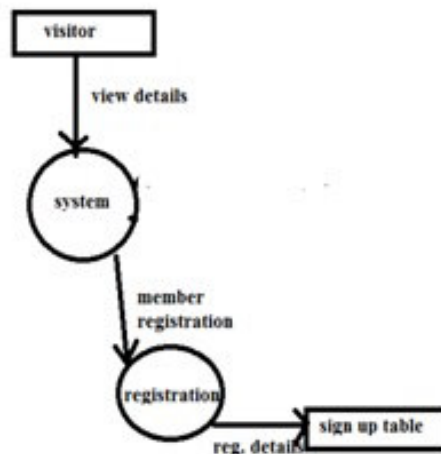


Fig.6.3.6 LEVEL 2 DFD VISITOR

## ER DIAGRAMS

### USE CASE VIEW

<b>Actor</b>	<b>Client and system</b>
<b>Goal</b>	Register, Place , review & cancel order, specify requirements
<b>Steps</b>	To be logged in Place an order or view the website or do enquiry Fill details Confirm action Logs out
<b>Result</b>	The system send the email or message for the confirmation

<b>Actor</b>	<b>owner and system</b>
<b>Goal</b>	Review the orders placed, provide the new information , schemes and offers, answer the queries of the clients
<b>Steps</b>	To be logged in Review the customer details Answer the queries if needed Give confirmation Logs out
<b>Result</b>	Can maintain the CRM . can interact with the clients
<b>Actor</b>	<b>Administrator &amp; system</b>
<b>Goal</b>	Do updations, managing profiles, handling database
<b>Steps</b>	No need to login
<b>Result</b>	Result into efficient handling of website.

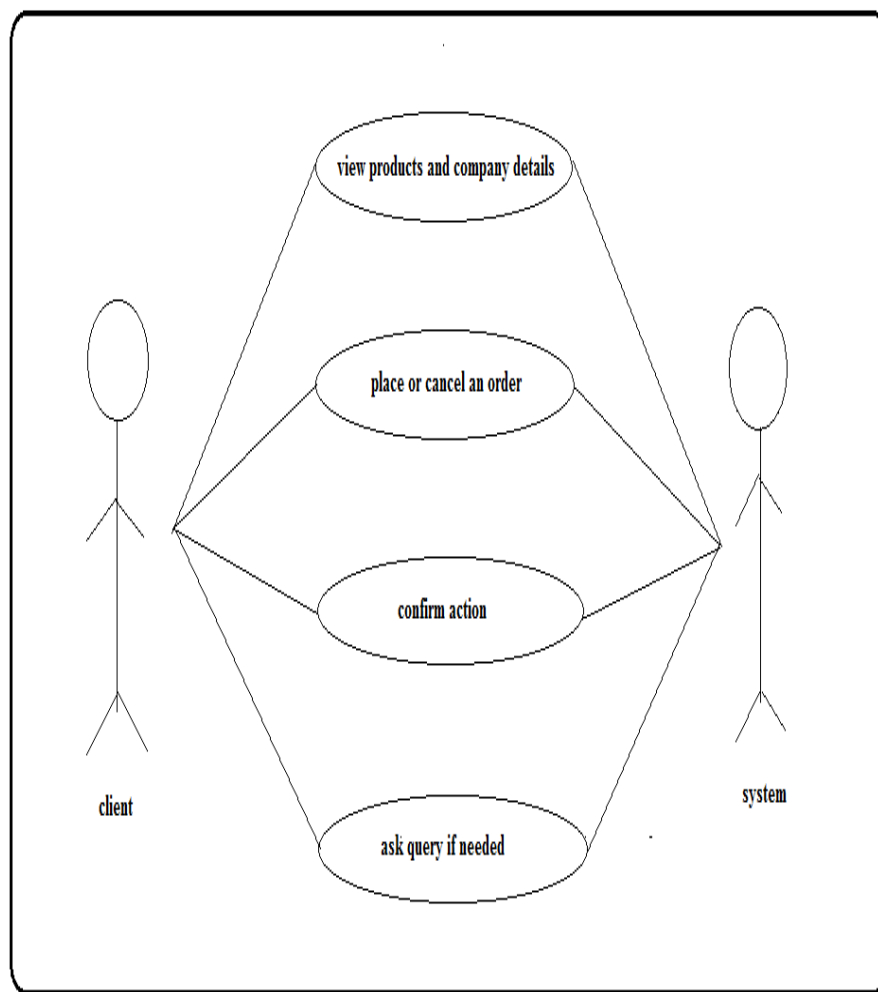


Fig. 6.3.7 Interaction of Client with System

## 6.4 FLOWCHARTS

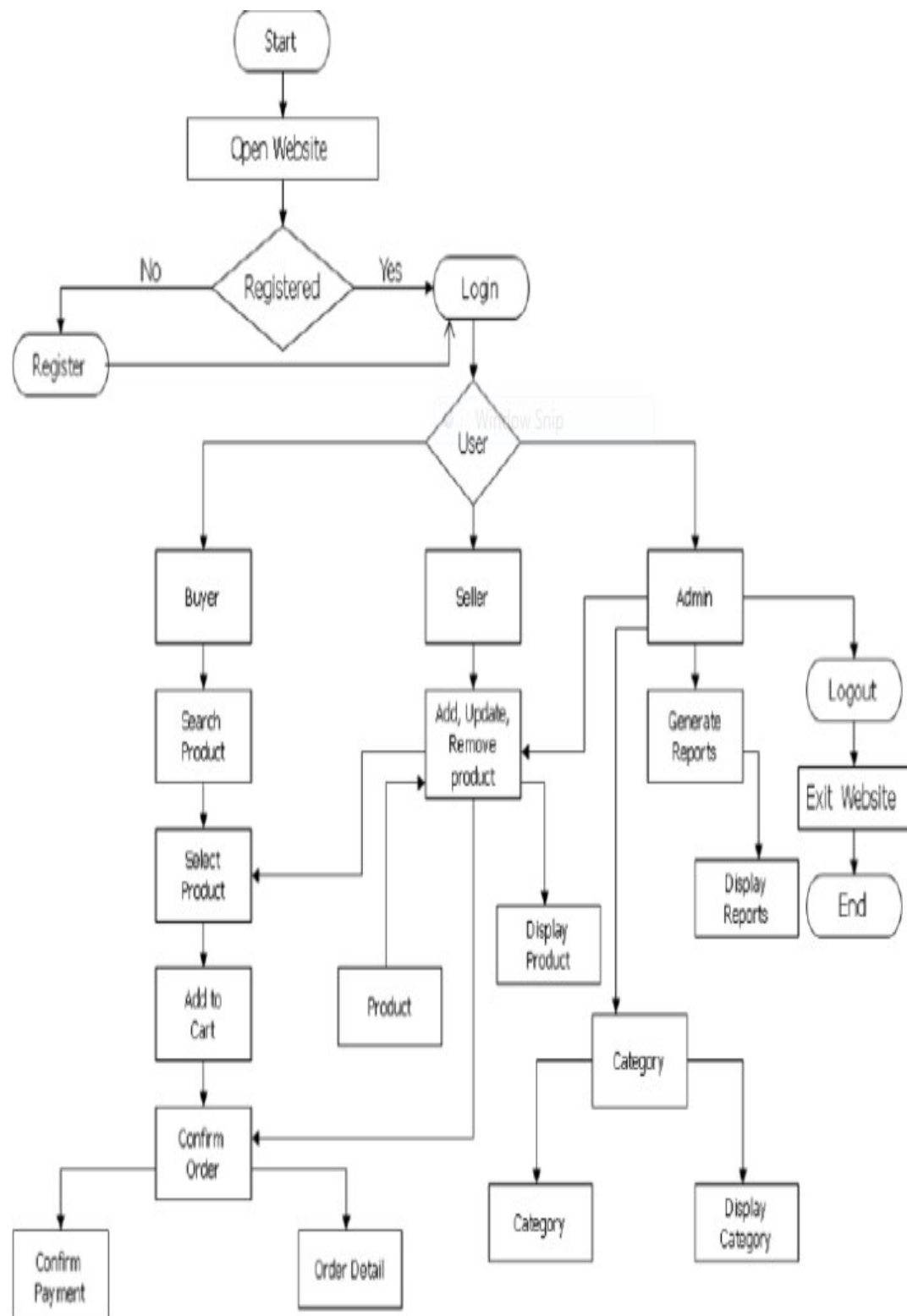


Fig. 6.4.1 Flowchart of Website

## 6.5 SERVER SIDE CODING

### MASTERPAGE

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Data.SqlClient;

public partial class MasterPage2 : System.Web.UI.MasterPage
{
    protected void Page_Load(object sender, EventArgs e)
    {
        // We check to see that this page loads by a new request for the page so that
        // this code does not
        // run when there is a postback like when a button is clicked.
        if (!Page.IsPostBack)
        {
            // We check to see if there exists an FormAuthenticationTicket for the
            // user.
            // Hide the "SigninPanel" and show the "CustomerPanel" if the user is
            // signed in
            // on this website.
            if (Request.IsAuthenticated == true)
            {
                // Get the FormsAuthentication ticket.
                FormsIdentity id = (FormsIdentity)Page.User.Identity;
                FormsAuthenticationTicket ticket = id.Ticket;

                // This is information from the FormsAuthenticationTicket
                lblUserName.Text = ticket.Name;

                // Set visibilities for panels
                SigninPanel.Visible = false;
                CustomerPanel.Visible = true;
                SmallCartPanel.Visible = true;

                // Call subroutine to load the small shopping cart
                LoadSmallCart();
            }
        }

        protected void cmdLogin_Click(object sender, System.EventArgs e)
        {
            // This code will get executed when the "Sign in" button is clicked and the
            // first thing that is
            // done is to check if the username and password corresponds to a user in the
            // "Customers" table.
            // We are sending the username and password to the "ValidateUser" function in
            // the If-statement
        }
    }
}
```

```

        // and will then return an answer from this function, se this function after
        this subroutine.
        if (ValidateUser(LoginUserName.Text, LoginUserPass.Text))
        {

            // We declare three variables that will be used to create a cookie for the
            signed in user.
            FormsAuthenticationTicket ticket = default(FormsAuthenticationTicket);
            string cookie = null;
            HttpCookie httpCookie = default(HttpCookie);

            // The FormsAuthenticationTicket is filled with data for version, name,
            issue date,
            // expiration date, if the cookie are persistent or not, user data and
            cookie path.
            ticket = new FormsAuthenticationTicket(1, LoginUserName.Text,
            DateTime.Now, DateTime.Now.AddMinutes(100), chkPersistCookie.Checked,
            HiddenCustomerID.Value, "MyPage");

            // The cookie is set to the encrypted ticket
            cookie = FormsAuthentication.Encrypt(ticket);

            // The httpCookie gets a name and the value from the cookie
            httpCookie = new HttpCookie(FormsAuthentication.FormsCookieName, cookie);

            // We set the expiration date for the httpCookie if the ticket is
            persistent. A HttpCookie
            // without an expire date will expire when the browser is closed.
            if (chkPersistCookie.Checked == true)
            {
                httpCookie.Expires = ticket.Expiration;
            }

            // Set the cookie path.
            httpCookie.Path = FormsAuthentication.FormsCookiePath;

            // Add the httpCookie.
            Response.Cookies.Add(httpCookie);

            // Redirect the user to the Default page so that the right panels are
            visible
            Response.Redirect("Default.aspx");

        }
        else
        {
            lblMessage.Text = "* Incorrect password or e-mail";
        }
    }

    private bool ValidateUser(string userName, string passWord)
    {
        // This is a check that is done for the entered user name, i has to be between
        1 and 80
        // characters. If this check is false this function will return a "False"
        statement and the
        // code below will not be executed. The "|" sign stands for "Or".
        if ((userName.Length == 0) | (userName.Length > 80))
        {
            System.Diagnostics.Trace.WriteLine("[ValidateUser] Input validation of
            userName failed.");
        }
    }

```



```

        return false;
    }

    // This is a check that is done for the entered password, it has to be between
    // 1 and 25
    // characters. If this check is false this function will return a "False"
    // statement and the
    // code below will not be executed. The "|" sign stands for "Or".
    if ((passWord.Length == 0) | (passWord.Length > 25))
    {
        System.Diagnostics.Trace.WriteLine("[ValidateUser] Input validation of
        passWord failed.");
        return false;
    }

    // We receive the userName and passWord as variables when this function is
    // called and we
    // create the variable "lookupPassword" as a string to store the password that
    // is selected from
    // the database. The hidden field "HiddenCustomer" that should store the
    // customer id of the customer
    // are set to Blank at the start. The passWord that is passed to this function
    // is encrypted with SHA1
    // because the password that is stored in the table "Customers" are encrypted
    // with SHA1 and therefore
    // have to do this encryption to compare the entered password with the stored
    // password in the database.
    string lookupPassword = null;
    HiddenCustomerID.Value = string.Empty;

    // Encrypt the password.
    string passwordHash =
    FormsAuthentication.HashPasswordForStoringInConfigFile(passWord, "SHA1");

    // This code is used to select the password and customer id from the
    // "Customers" table according to
    // the supplied username in the "sign in form".

    // Declare variables for a connection string and a SELECT statement.
    string ConnString =
    ConfigurationManager.ConnectionStrings["ConnectionString"].ToString();
    string SelectUser = "SELECT CustomerID, Password FROM Customers WHERE Username
    = @Username";

    // Create a SqlConnection. The using block is used to call dispose (close)
    // automatically even
    // if there are an exception.
    using (SqlConnection cn = new SqlConnection(ConnString))
    {
        // Create a SqlCommand.
        SqlCommand cmd = new SqlCommand(SelectUser, cn);

        // Create a SqlDataReader.
        SqlDataReader reader = null;

        // Add parameters.
        cmd.Parameters.AddWithValue("@Username", LoginUserName.Text);

        // The Try/Catch/Finally block is used to handle exceptions.
        try
        {
            // Open the connection.

```

```

        cn.Open();

        // We use SqlDataReader and just want to select one single row. The
CustomerID are // supplied to the hidden field "HiddenCustomerID" and the Password
are supplied to the // "lookupPassword" string.

        // Execute the SELECT statement and fill the reader with data.
        reader = cmd.ExecuteReader(CommandBehavior.SingleRow);

        // Loop the reader.
        while (reader.Read())
        {
            HiddenCustomerID.Value = reader["CustomerID"].ToString();
            lookupPassword = reader["Password"].ToString();
        }
    }
    catch (Exception ex)
    {
        System.Diagnostics.Trace.WriteLine("[ValidateUser] Exception " +
ex.Message);
    }
    finally
    {
        // Dispose the SqlCommand.
        cmd.Dispose();

        // Close the reader.
        if (reader != null)
            reader.Close();
    }
}

// If no password is found this function will return false.
if (lookupPassword == null)
{
    // You could write failed login attempts here to the event log for
additional security.
    return false;
}

// Compare lookupPassword and passwordHash by using a case-sensitive
comparison.
return (string.Compare(lookupPassword, passwordHash, false) == 0);
}

protected void btnSignOut_Click(object sender, System.EventArgs e)
{
    // This code is executed when the user clicks the "Sign out" button and
deletes the cookie and the ticket
    // for the customer.
    FormsAuthentication.SignOut();
    FormsAuthentication.RedirectToLoginPage();
}

public void LoadSmallCart()
{
    // We declare "num" to iterate through the HttpCookie and we declare
"CartCookie" to be able to check for
    // its existence and to get data from this HttpCookie.
    Int32 num = default(Int32);

```

```

        HttpCookie CartCookie = Request.Cookies.Get("CartCookie");
        ltCartItem.Text = string.Empty;

        // We check to see if the HttpCookie with the name of CartCookie exists so
        that we not
        // will get any null point exeptions if the HttpCookie does not exist. We also
        check if
        // the "CartCookie" has any keys, if there are no keys there are no products
        in the
        // shopping cart and then we create a message for this.
        if (CartCookie != null)
        {
            if (CartCookie.HasKeys)
            {
                // We iterate through each post in the HttpCookie and get the
                ProductID and Quantity for
                // each post. We update the literal control "ltCartItem" with data
                from each row in the HttpCookie.
                for (num = 0; num <= CartCookie.Values.Count - 1; num++)
                {
                    ltCartItem.Text += CartCookie.Values.AllKeys[num] + " (" +
                    CartCookie.Values[num] + ")" + "<br />";
                }
            }
            else
            {
                ltCartItem.Text = "There are no items in your shopping cart.";
            }
        }
        else
        {
            ltCartItem.Text = "There are no items in your shopping cart.";
        }
    }
}

```

## ORDER PAGE

```

using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Data.SqlClient;
public partial class Default2 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        // We check to see that this page loads without any postback on the page
        because it is unnecessary to
        // to run this code if there only is a postback on the page.
        if (!Page.IsPostBack)
        {

```

```

        // We check to see if there exists an FormAuthenticationTicket for the
user.
        if (Request.IsAuthenticated == true)
        {
            // Create a FormsIdentity and a ticket.
            FormsIdentity id = (FormsIdentity)User.Identity;
            FormsAuthenticationTicket ticket = id.Ticket;

            // The UserData information from the FormsAuthenticationTicket
contains the
            // "CustomerID" of the signed in user and this value is stored in the
hidden field "HiddenCustomerID".
            HiddenCustomerID.Value = ticket.UserData;

            // We call this subroutine to create the order list
            LoadOrderList();
        }
        else
        {
            Response.Redirect("Not-signed-in.aspx");
        }
    }

protected void LoadOrderList()
{
    // We select all orders from the "Orders" table where the CustomerID equals
the CustomerID stored in the
    // hidden field "HiddenCustomerID" and binds the data to the
"OrderListRepeater" control.

    // Declare variables for a connection string and a SELECT statement.
    string ConnString =
ConfigurationManager.ConnectionStrings["ConnectionString"].ToString();
    string sql = "SELECT * FROM Orders WHERE CustomerID = @CustomerID ORDER BY
OrderID ASC";

    // Create a SqlConnection. The using block is used to call dispose (close)
automatically even
    // if there are an exception.
    using (SqlConnection cn = new SqlConnection(ConnString))
    {
        // Create a SqlCommand.
        SqlCommand cmd = new SqlCommand(sql, cn);

        // Create a SqlDataReader.
        SqlDataReader reader = null;

        // Add parameters.
        cmd.Parameters.AddWithValue("@CustomerID", HiddenCustomerID.Value);

        // The Try/Catch/Finally block is used to handle exceptions.
        try
        {
            // Open the connection.
            cn.Open();

            // Execute the SELECT statement and fill the reader with data.
            reader = cmd.ExecuteReader();
        }
    }
}

```

```

        // Bind data to the OrderListRepeater with the reader as datasource.
        OrderListRepeater.DataSource = reader;
        OrderListRepeater.DataBind();

    }
    catch (Exception ex)
    {
        Response.Write(ex.Message);
    }
    finally
    {
        // Dispose the SqlCommand.
        cmd.Dispose();

        // Close the reader.
        if (reader != null)
            reader.Close();
    }
}

protected void OrderListRepeater_ItemCommand(object source,
System.Web.UI.WebControls.RepeaterCommandEventArgs e)
{
    // We pick up the order id for the selected order according to the command
    argument and store
    // this value in a variable called "OrderID"
    string OrderID = (e.CommandArgument).ToString();

    // We set visibility for OrderListPanel and OrderPanel.
    OrderListPanel.Visible = false;
    OrderPanel.Visible = true;

    // We select the order from the Orders table that has the same OrderID as the
    value in the
    // OrderID variable. We use an SQL datareader to get data and passes the data
    from one single row
    // (CommandBehavior.SingleRow) to labels on the Orders.aspx webpage.

    // Declare variables for a connection string and a SELECT statement.
    string ConnString1 =
ConfigurationManager.ConnectionStrings["ConnectionString"].ToString();
    string sql1 = "SELECT * FROM Orders WHERE OrderID = @OrderID";

    // Create a SqlConnection. The using block is used to call dispose (close)
    automatically even
    // if there are an exception.
    using (SqlConnection cn = new SqlConnection(ConnString1))
    {
        // Create a SqlCommand.
        SqlCommand cmd = new SqlCommand(sql1, cn);

        // Create a SqlDataReader.
        SqlDataReader reader = null;

        // Add parameters.
        cmd.Parameters.AddWithValue("@OrderID", OrderID);

        // The Try/Catch/Finally block is used to handle exceptions.
        try
        {

```

```

        // Open the connection.
        cn.Open();

        // Execute the SELECT statement and fill the reader with data.
        reader = cmd.ExecuteReader(CommandBehavior.SingleRow);

        // Loop the reader.
        while (reader.Read())
        {
            lbOrderID.Text = reader["OrderID"].ToString();
            lbOrderDate.Text = string.Format("{0:yyyy-MM-dd}",
reader["OrderDate"].ToString());
            lbName.Text = reader["Company"].ToString();
            lbCustomerNumber.Text = reader["CustomerID"].ToString();
            lbAttention.Text = reader["Attention"].ToString();
            lbContact.Text = reader["Contact"].ToString();
            lbAddress.Text = reader["Adress"].ToString();
            lbPostalCode.Text = reader["PostalCode"].ToString();
            lbCity.Text = reader["City"].ToString();
            lbCountry.Text = reader["Country"].ToString();
        }

    }
    catch (Exception ex)
    {
        Response.Write(ex.Message);
    }
    finally
    {
        // Dispose the SqlCommand.
        cmd.Dispose();

        // Close the reader.
        if (reader != null)
            reader.Close();
    }
}

// When we have selected a order we want to select the product rows for this
order in the OrdersProducts
// table and fill the "ProductRowRepeater" with these rows. In our SELECT
statement we have a INNER JOIN
// statement to get the product name from the "Products" table and a
calculation for row sum.

// Declare variables for a connection string and a SELECT statement.
string ConnString2 =
ConfigurationManager.ConnectionStrings["ConnectionString"].ToString();
string sql2 = "SELECT O.ProductID, P.ProductName , O.Quantity,
O.PriceExSaleTax FROM OrdersProducts As O INNER JOIN Products As P ON P.ProductID =
O.ProductID WHERE OrderID = @OrderID GROUP BY O.ProductID, P.ProductName, O.Quantity,
O.PriceExSaleTax ORDER BY ProductID ASC";

// Create a SqlConnection. The using block is used to call dispose (close)
automatically even
// if there are an exception.
using (SqlConnection cn = new SqlConnection(ConnString2))
{
    // Create a SqlCommand.
    SqlCommand cmd = new SqlCommand(sql2, cn);

    // Create a SqlDataReader.

```

```

        SqlDataReader reader = null;

        // Add parameters.
        cmd.Parameters.AddWithValue("@OrderID", OrderID);

        // The Try/Catch/Finally block is used to handle exceptions.
        try
        {
            // Open the connection.
            cn.Open();

            // Execute the SELECT statement and fill the reader with data.
            reader = cmd.ExecuteReader();

            // Bind data to the ProductRowRepeater.
            ProductRowRepeater.DataSource = reader;
            ProductRowRepeater.DataBind();

        }
        catch (Exception ex)
        {
            Response.Write(ex.Message);
        }
        finally
        {
            // Dispose the SqlCommand.
            cmd.Dispose();

            // Close the reader.
            if (reader != null)
                reader.Close();
        }
    }

    // Call subroutine to calculate total sums.
    CalculateOrderSums();
}

protected void linkOrderList_Click(object sender, System.EventArgs e)
{
    // Set visibility for OrderListPanel and OrderPanel.
    OrderListPanel.Visible = true;
    OrderPanel.Visible = false;
}

protected void CalculateOrderSums()
{
    // We declare two variables that we will use for our calculations of total
    sums.
    decimal PriceExVat = 0m;
    decimal VatMoney = 0m;

    // We iterate through each row in the "ProductRowRepeater and add values to
    our two variables
    foreach (RepeaterItem RepeaterRow in ProductRowRepeater.Items)
    {
        //Literal VatObj = (Literal)RepeaterRow.FindControl("ltVAT");
        Literal RowSumObj = (Literal)RepeaterRow.FindControl("ltRowSum");

        //PriceExVat += Convert.ToDecimal(RowSumObj.Text);
    }
}

```

```

        //VatMoney += Convert.ToDecimal(RowSumObj.Text) *
        (Convert.ToDecimal(VatObj.Text) / 100);
    }

    // We add the sums to labels and calculate the totalsum as price excluding VAT
    plus VAT in money
    lblPriceTotal.Text = Convert.ToString(PriceExVat);
    //lblVatTotal.Text = Convert.ToString(VatMoney);
    lblTotalSum.Text = Convert.ToString(PriceExVat + VatMoney);

}
}

```

## PRODUCT PAGE

```

using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Data.SqlClient;

public partial class Default2 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        // We check to see that this page loads without any postback on the page
        because it is unnecessary to
        // to run this code if there only is a postback on the page like a click on a
        button.
        if (!Page.IsPostBack)
        {
            // When this webpage loads we request the product id from the parameter
            "Pid" in the url and
            // places this value in the hidden field "HiddenProductID".
            HiddenProductID.Value = Convert.ToString(Request.QueryString["Pid"]);

            // We select the product that has the product id that is saved in the
            "HiddenProductID" field and
            // binds the data from the select statement to the controls on
            "Product.aspx".

            // Declare variables for a connection string and a SELECT statement.
            string ConnString =
            ConfigurationManager.ConnectionStrings["ConnectionString"].ToString();
            string sql = "SELECT ProductID, ProductName, Description, PriceExSaleTax
            FROM Products WHERE ProductID = @ProductID";

            // Create a SqlConnection. The using block is used to call dispose (close)
            automatically even
            // if there are an exception.
            using (SqlConnection cn = new SqlConnection(ConnString))

```



```

{
    // Create a SqlCommand.
    SqlCommand cmd = new SqlCommand(sql, cn);

    // Create a SqlDataReader.
    SqlDataReader reader = null;

    // Add parameters.
    cmd.Parameters.AddWithValue("@ProductID", HiddenProductID.Value);

    // The Try/Catch/Finally block is used to handle exceptions.
    try
    {
        // Open the connection.
        cn.Open();

        // Execute the select statement and fill the reader.
        reader = cmd.ExecuteReader(CommandBehavior.SingleRow);

        // Loop the reader.
        while (reader.Read())
        {
            HiddenProductID.Value = reader["ProductID"].ToString();
            ltProductName.Text = reader["ProductName"].ToString();
            ltDescription.Text = reader["Description"].ToString();
            ltPriceExSaleTax.Text = reader["PriceExSaleTax"].ToString();
            ltSaleTax.Text = reader["SaletaxMoney"].ToString();
            ltSaleTaxPercent.Text = string.Format("{0:P}",
reader["SaletaxPercent"]);
            ltTotalPrice.Text = reader["TotalPrice"].ToString();
        }
    }
    catch (Exception ex)
    {
        Response.Write(ex.Message);
    }
    finally
    {
        // Dispose the SqlCommand.
        cmd.Dispose();

        // Close the reader.
        if (reader != null)
            reader.Close();
    }
}

protected void btnBuy_Click(object sender, System.EventArgs e)
{
    // We first check to see if the user has signed in and if the user not is
    signed in we redirect him
    // to the "Not-signed-in.aspx" webpage.
    if (Request.IsAuthenticated == true)
    {
        // We declare one variable as int32 to be able to use the value in the
        quantity textbox.
        Int32 AddNumberOfUnits = default(Int32);

        // We try to convert the text in the textbox "txtQuantity" to a Int32 with
        the use of TRYPARSE and if this

```

```

        // operation succeed this value will be added to the "AddNumberOfUnits"
variable. If the conversion not is
        // a success the "AddNumberOfUnits" variable gets 0 as the value.
        Int32.TryParse(txtQuantity.Text, out AddNumberOfUnits);

        // We declare ShoppingCookie as a HttpCookie, selects the cookie with the
name "CartCookie" and
        // store this cookie in "ShoppingCookie". If we don't find any cookie with
the name of "CartCookie"
        // "ShoppingCookie will get "Nothing" as the value.
        HttpCookie ShoppingCookie = default(HttpCookie);
        ShoppingCookie = Request.Cookies.Get("CartCookie");

        // We check to see if the HttpCookie with the name of CartCookie exists so
that we not
        // will get any null point exeptions if the HttpCookie does not exist. If
the cookie
        // exists we use this cookie and if the cookie does not exist we create a
new cookie.
        if (ShoppingCookie != null)
        {
            // We check to see if the HttpCookie has keys, if the HttpCookie has
keys we
            // find the value (quantity) from the key that corresponds to the
ProductID
            // and add the supplied quantity to the old quantity. If the
HttpCookie does
            // not have keys we just add the supplied quantity to a key with the
ProductID and
            // deletes on key with the value of "Nothing" that are created in this
case.
            if (ShoppingCookie.HasKeys)
            {
                Int32 OldQty =
Convert.ToInt32(Request.Cookies["CartCookie"][HiddenProductID.Value]);

                ShoppingCookie.Values[HiddenProductID.Value] =
Convert.ToString(OldQty + AddNumberOfUnits);
                ShoppingCookie.Expires = DateTime.Now.AddHours(3);
                Response.Cookies.Add(ShoppingCookie);
            }
            else
            {
                ShoppingCookie.Values[HiddenProductID.Value] =
Convert.ToString(AddNumberOfUnits);
                ShoppingCookie.Values.Remove(null);
                ShoppingCookie.Expires = DateTime.Now.AddHours(3);
                Response.Cookies.Add(ShoppingCookie);
            }
        }
        else
        {
            // If a CartCookie does not exist we create a new HttpCookie and add
the
            // quantity to a key with the name of the ProductID.
            ShoppingCookie = new HttpCookie("CartCookie");
            ShoppingCookie.Values[HiddenProductID.Value] =
Convert.ToString(AddNumberOfUnits);
            ShoppingCookie.Expires = DateTime.Now.AddHours(3);
            Response.Cookies.Add(ShoppingCookie);
        }
    }

```

```

        // Update the small shopping cart by calling a public method in the Start
class on
    // the masterpage.
    ((MasterPage2)this.Master).LoadSmallCart();

    }
    else
    {
        Response.Redirect("Not-signed-in.aspx");
    }
}

}

using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Data.SqlClient;

public partial class Default2 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        // We check to see that this page loads without any postback on the page
        // because it is unnecessary to
        // to run this code if there only is a postback on the page like a click on a
        // button.
        if (!Page.IsPostBack)
        {
            // We select all products from the "Products" table and sort the list by
            // "ProductName" in ascending order.
            // The data for the products are supplied to the "ProductListRepeater"
            // repeater control and we split up the
            // product list in pages with next and previous links for navigation.

            // Declare variables for a connection string and a SELECT statement.
            string ConnString =
            ConfigurationManager.ConnectionStrings["ConnectionString"].ToString();
            string sql = "SELECT ProductID, ProductName, Description, PriceExSaleTax
            FROM Products ORDER BY ProductName ASC";

            // Create a SqlConnection. The using block is used to call dispose (close)
            // automatically even
            // if there are an exception.
            using (SqlConnection cn = new SqlConnection(ConnString))
            {
                // Create a SqlDataAdapter.
                SqlDataAdapter sad = new SqlDataAdapter();

                // Create a SqlCommand.
                SqlCommand cmd = new SqlCommand(sql, cn);

                // Create a DataTable.
                DataTable dt = new DataTable();

```

```

// Create a PagedDataSource.
PagedDataSource objPds = new PagedDataSource();

// The Try/Catch/Finally block is used to handle exceptions.
try
{
    // Open the SqlConnection.
    cn.Open();

    // Assign the select command to the SqlDataAdapter.
    sad.SelectCommand = cmd;

    // Fill the data table "dt" with data from the SqlDataAdapter
    sad.Fill(dt);

    // Populate the repeater control with the datatable
    objPds.DataSource = dt.DefaultView;

    // Indicate that the data should be paged
    objPds.AllowPaging = true;

    // Set the pagesize
    objPds.PageSize = 10;

    // The current page (curpage) is declared as an integer
    Int32 curpage;

    // We set the page number for the current page from the page
parameter in the url
    // or to 1 if there are no page parameter in the url
    if (Request.QueryString["page"] != null)
    {
        curpage = Convert.ToInt32(Request.QueryString["page"]);
    }
    else
    {
        curpage = 1;
    }

    // The current page index is equal to the current page minus 1 and
we need to know this
    objPds.CurrentPageIndex = curpage - 1;

    // We set the text in the middle to show the current pagenumber
and the last pagenumber
    lblCurrentPage.Text = "| Page: " + curpage.ToString() + " of " +
objPds.PageCount.ToString() + "|";

    // This is code for the link to the previous page, we don't show
this link on the first page
    if (!objPds.IsFirstPage)
    {
        lnkPrev.Visible = true;
        lnkPrev.NavigateUrl = "~/Home.aspx?" + "page=" +
Convert.ToString(curpage - 1);
    }

    // This is code for the link to the next page, we don't show this
link on the last page
    if (!objPds.IsLastPage)

```

```

        {
            lnkNext.Visible = true;
            lnkNext.NavigateUrl = "~/Home.aspx?" + "page=" +
Convert.ToString(curpage + 1);
        }

        // This code is used to bind data to the repeater control.
        ProductListRepeater.DataSource = objPds;
        ProductListRepeater.DataBind();

    }
    catch (Exception ex)
    {
        Response.Write(ex.Message);
    }
    finally
    {
        // Dispose of objects to avoid memory leakage.
        sad.Dispose();
        cmd.Dispose();
        dt.Dispose();
    }
}
}
}

protected string GenerateURL(object Product)
{
    //Create a URL for each product link
    string strProdUrl = "~/Product.aspx?Pid=" + Product;

    return strProdUrl;
}
}

```

PRODUCTCART PAGE

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class Default2 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }
    protected string GenerateURL(object Product)
    {
        //Create a URL for each product link
        string strProdUrl = "~/Product.aspx?Pid=" + Product;

        return strProdUrl;
    }
}

```

REGISTER PAGES

```

using System;

```

```

using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Data.SqlClient;

public partial class Default2 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }
    protected void btnRegister_Click(object sender, EventArgs e)
    {
        // Takes the data from the form at "Register-customer.aspx" when the button is
        // clicked and then inserts
        // this data to the table "Customers" in the "Webshop" database. The password
        // is encrypted with SHA1 and stored
        // as an encrypted password in the table.

        // Set the SqlEx label to a empty string.
        SqlEx.Text = string.Empty;

        string passwordHash =
FormsAuthentication.HashPasswordForStoringInConfigFile(txtPassword.Text, "SHA1");
        string ConnString =
ConfigurationManager.ConnectionStrings["ConnectionString"].ToString();
        string sql = "INSERT INTO Customers (Username, Password, Company, OrgNumber,
Contact, Attention, Address, "
+ "PostalCode, City, Country) VALUES (@Username, @Password, @Company,
@OrgNumber, @Contact, @Attention, "
+ "@Address, @PostalCode, @City, @Country)";

        // Create a SqlConnection. The using block is used to call dispose (close)
        // automatically even
        // if there are an exception.
        using (SqlConnection cn = new SqlConnection(ConnString))
        {
            // Create a SqlCommand.
            SqlCommand cmd = new SqlCommand(sql, cn);

            // Add parameters.
            cmd.Parameters.AddWithValue("@Username", txtUserName.Text);
            cmd.Parameters.AddWithValue("@Password", passwordHash);
            cmd.Parameters.AddWithValue("@Company", txtCompanyName.Text);
            cmd.Parameters.AddWithValue("@OrgNumber", txtOrganisationNumber.Text);
            cmd.Parameters.AddWithValue("@Contact", txtContact.Text);
            cmd.Parameters.AddWithValue("@Attention", txtAttention.Text);
            cmd.Parameters.AddWithValue("@Address", txtAddress.Text);
            cmd.Parameters.AddWithValue("@PostalCode", txtPostalCode.Text);
            cmd.Parameters.AddWithValue("@City", txtCity.Text);
            cmd.Parameters.AddWithValue("@Country", txtCountry.Text);

            // The Try/Catch/Finally block is used to handle exceptions.
            try
            {
                // Open the connection.

```

```

        cn.Open();

        // Execute the INSERT statement.
        cmd.ExecuteNonQuery();

        // Set visibility for panels.
        CurrentPanel.Visible = false;
        ThankYouPanel.Visible = true;
    }
    catch (SqlException Sqlexc)
    {
        switch (Sqlexc.Number)
        {
            case 2601:
                SqlEx.Text = "* Given E-mail already exists";
                break;
            default:
                Response.Write(Sqlexc.Message);
                break;
        }
    }
    catch (Exception ex)
    {
        Response.Write(ex.Message);
    }
    finally
    {
        // Dispose the SqlCommand to avoid memory leakage.
        cmd.Dispose();
    }
}
}
}

```

#### SHOPPING CART PAGE

```

using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Data.SqlClient;

public partial class Default2 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        // We check to see that this page loads without any postback on the page
        // because it is unnecessary to
        // to run this code if there only is a postback on the page like a click on a
        // button.
        if (!Page.IsPostBack)
        {
            // We check to see if there exists an FormAuthenticationTicket for the
            // user.

```

```

        if (Request.IsAuthenticated == true)
        {
            // Declare variables for a FormsIdentity and a ticket.
            FormsIdentity id = (FormsIdentity)User.Identity;
            FormsAuthenticationTicket ticket = id.Ticket;

            // We get the CustomerID from UserData in the
            FormsAuthenticationTicket and store this value
            // in the hidden field "HiddenCustomerID".
            HiddenCustomerID.Value = Convert.ToString(ticket.UserData);

            // We call this subroutine to create our shopping cart.
            LoadShoppingCart();
        }
        else
        {
            Response.Redirect("Not-signed-in.aspx");
        }
    }

    protected void LoadShoppingCart()
    {
        // We create a DataTable (in memory) to store data in and a DataRow so that we
        can add rows to the
        // DataTable as we iterate through each product in the HttpCookie.
        DataTable tblShoppingCart = new DataTable("CartTable");
        DataRow rwShoppingCart = null;

        // We declare "num" to iterate through the HttpCookie and we declare
        "CartCookie" to be able to check for
        // its existence and to get data from this HttpCookie.
        Int32 num = default(Int32);
        HttpCookie CartCookie = Request.Cookies.Get("CartCookie");

        // We check to see if the HttpCookie with the name of CartCookie exists so
        that we not
        // will get any null point exceptions if the HttpCookie does not exist. We also
        check if
        // the "CartCookie" has any keys, if there are no keys there are no products
        in the
        // shopping cart and then we will get an exception if we try to select
        products
        // from the "Products" table.
        if (CartCookie != null)
        {
            if (CartCookie.HasKeys)
            {
                // We add 5 columns to our DataTable, we need to add the columns
                before we iterate through each
                // product so that we get only one set of columns :-).
                tblShoppingCart.Columns.Add("ProductIDC");
                tblShoppingCart.Columns.Add("ProductName");
                //tblShoppingCart.Columns.Add("SaletaxPercent");
                tblShoppingCart.Columns.Add("QuantityC");
                tblShoppingCart.Columns.Add("PriceExSaleTax");

                // We iterate through each post in the HttpCookie and get the
                ProductID and Quantity for
                // each post. We use the ProductID to select data from the Products
                table.
            }
        }
    }

```



```

for (num = 0; num <= CartCookie.Values.Count - 1; num++)
{
    string ProductID = CartCookie.Values.AllKeys[num];
    string Quantity = CartCookie.Values[num];

    try
    {
        string ConnString =
ConfigurationManager.ConnectionStrings["ConnectionString"].ToString();
        string sql = "SELECT * FROM Products WHERE ProductID =
@ProductID";

        // Create a SqlConnection. The using block is used to call
dispose (close) automatically
        // even if there are an exception.
        using (SqlConnection cn = new SqlConnection(ConnString))
        {
            // Create a SqlCommand. The using block is used to call
dispose (close) automatically
            // even if there are an exception.
            using (SqlCommand cmd = new SqlCommand(sql, cn))
            {
                // Add parameters.
                cmd.Parameters.AddWithValue("@ProductID", ProductID);

                // Open the connection.
                cn.Open();

                // Create a SqlDataReader and execute the SELECT
command. The using block is used
                // to call dispose (close) automatically even if there
are an exception.
                using (SqlDataReader reader =
cmd.ExecuteReader(CommandBehavior.SingleRow))
                {
                    while (reader.Read())
                    {
                        // We add one row to the DataTable and fill
each cell with data.

                        rwShoppingCart = tblShoppingCart.NewRow();
                        rwShoppingCart[0] = ProductID;
                        rwShoppingCart[1] =

reader["ProductName"].ToString();
                        //rwShoppingCart[2] =
Convert.ToDecimal(reader["SaletaxPercent"]) * 100;
                        rwShoppingCart[2] = Quantity;
                        rwShoppingCart[3] =

reader["PriceExSaleTax"].ToString();
                        tblShoppingCart.Rows.Add(rwShoppingCart);
                    }
                }
            }
        }
    }
    catch (Exception ex)
    {
        Response.Write(ex.Message);
    }
}

```

```

    }

    // We fill our Repeater control with the data from the
    "tblShoppingCart" DataTable.
    CartRepeater.DataSource = tblShoppingCart;
    CartRepeater.DataBind();

    // Call a subroutine to calculate totals in the shopping cart.
    CalculateCartSums();

}
else
{
    // If the HttpCookie does not have any keys we update the shopping
    cart so that it does not show
    // any rows.
    CartRepeater.DataSource = string.Empty;
    CartRepeater.DataBind();

    // Call a subroutine to calculate totals in the shopping cart
    CalculateCartSums();

}
}

protected void CartRepeater_ItemCommand(object source,
System.Web.UI.WebControls.RepeaterCommandEventArgs e)
{
    // This code runs when a user has clicked a remove button (X) on a row.
    // We get the ProductID from the CommandArgument when the remove button is
    clicked.
    string ProductID = (e.CommandArgument).ToString();

    // We declare the variable CartCookie to check if the HttpCookie "CartCookie"
    exists and to
    // be able to remove a key from it.
    HttpCookie CartCookie = Request.Cookies.Get("CartCookie");

    // We check if the CartCookie exists and then removes the key that has the
    "ProductID" as its name
    // and then we then reload the shopping cart so that our repeater gets
    updated.
    if (CartCookie != null)
    {
        CartCookie.Values.Remove(ProductID);
        CartCookie.Expires = DateTime.Now.AddHours(3);
        Response.Cookies.Add(CartCookie);

        // Reload the shopping cart
        LoadShoppingCart();

        // Update the small shopping cart by calling a public method in the Start
        class for the masterpage.
        ((MasterPage2)this.Master).LoadSmallCart();
    }
}

protected void btnUpdateCart_Click(object sender, System.EventArgs e)
{

```

```

        // We declare the variable CartCookie to check if the HttpCookie "CartCookie"
        exists and to
        // be able to remove a key from it.
        HttpCookie CartCookie = Request.Cookies.Get("CartCookie");

        // We check to see if the HttpCookie with the name of CartCookie exists so
        that we not
        // will get any null point exeptions if the HttpCookie does not exist.
        if (CartCookie != null)
        {
            if (CartCookie.HasKeys)
            {
                foreach (RepeaterItem RepeaterRow in CartRepeater.Items)
                {
                    Literal ProductObj =
                    (Literal)RepeaterRow.FindControl("ltProductID");
                    TextBox QuantityObj =
                    (TextBox)RepeaterRow.FindControl("txtQuantity");

                    CartCookie.Values[ProductObj.Text] = QuantityObj.Text;
                    CartCookie.Expires = DateTime.Now.AddHours(3);
                    Response.Cookies.Add(CartCookie);
                }
            }

            // Recalculate the shopping cart
            CalculateCartSums();

            // Update the small shopping cart by calling a public method in the Start
            class for the masterpage.
            ((MasterPage2)this.Master).LoadSmallCart();
        }
    }

    protected void btnCheckOut_Click(object sender, System.EventArgs e)
    {
        // When the user wants to check out and has clicked the "Check out" button we
        make the
        // "CheckOutPanel" (includes textboxes for customer information) visible.
        CheckOutPanel.Visible = true;

        // We use the value for CustomerID that is stored in the hidden field
        "HiddenCustomerID" to get
        // information on the customer in the "Customers" table and inserts this
        information in textboxes.
        // We use an SqlDataReader to get the information and just want to select one
        single row (CommandBehavior.SingleRow).

        try
        {
            string ConnString =
            ConfigurationManager.ConnectionStrings["ConnectionString"].ToString();
            string sql = "SELECT * FROM Customers WHERE CustomerID = @CustomerID";

            // Create a SqlConnection. The using block is used to call dispose (close)
            automatically even
            // if there are an exception.
            using (SqlConnection cn = new SqlConnection(ConnString))
            {

```

```

        // Create a SqlCommand. The using block is used to call dispose
(close) automatically even
        // if there are an exception.
        using (SqlCommand cmd = new SqlCommand(sql, cn))
        {
            // Add parameters.
            cmd.Parameters.AddWithValue("@CustomerID",
HiddenCustomerID.Value);

            // Open the connection.
            cn.Open();

            // Create a SqlDataReader. The using block is used to call dispose
(close) automatically
            // even if there are an exception.
            using (SqlDataReader reader =
cmd.ExecuteReader(CommandBehavior.SingleRow))
            {
                while (reader.Read())
                {
                    txtCompany.Text = reader["Company"].ToString();
                    txtOrganisationNumber.Text =
reader["OrgNumber"].ToString();
                    txtContact.Text = reader["Contact"].ToString();
                    txtAttention.Text = reader["Attention"].ToString();
                    txtAddress.Text = reader["Address"].ToString();
                    txtPostalCode.Text = reader["PostalCode"].ToString();
                    txtCity.Text = reader["City"].ToString();
                    txtCountry.Text = reader["Country"].ToString();
                }
            }
        }
    }
    catch (Exception ex)
    {
        Response.Write(ex.Message);
    }
}

protected void btnOrder_Click(object sender, System.EventArgs e)
{
    //// We declare the variable "Identity" that will store the OrderID of the
order that is added so that we can use
    // this OrdeID number when we are to insert data in the table "OrderProduct"
that has a many to many relationship
    // to the Order table. Check database diagrams to se the relationships between
tables.

    Int64 Identity = 0;

    // We insert an order in the "Orders" table that has OrderID as a identity
field that increments with 1.
    // We have added ; SELECT SCOPE_IDENTITY() to the SQL statement to get the
OrderID of the inserted order
    // and use "ExecuteScalar()" instead of "ExecuteNonQuery()" in the SQL
command.

    try
    {

```

```

        string ConnString =
ConfigurationManager.ConnectionStrings["ConnectionString"].ToString();
        string sql = "INSERT INTO Orders (OrderDate, CustomerID, Company,
OrgNumber, Contact, Attention, "
+ "Address, PostalCode, City, Country) VALUES (@OrderDate, @CustomerID,
@Company, @OrgNumber, "
+ "@Contact, @Attention, @Address, @PostalCode, @City, @Country); SELECT
SCOPE_IDENTITY()";

        // Create a SqlConnection. The using block is used to call dispose (close)
automatically even if
        // there are an exception.
        using (SqlConnection cn = new SqlConnection(ConnString))
        {
            // Create a SqlCommand. The using block is used to call dispose
(close) automatically even if
            // there are an exception.
            using (SqlCommand cmd = new SqlCommand(sql, cn))
            {
                // Add parameters.
                cmd.Parameters.AddWithValue("@OrderDate",
DateTime.Now.ToString());
                cmd.Parameters.AddWithValue("@CustomerID",
HiddenCustomerID.Value);
                cmd.Parameters.AddWithValue("@Company", txtCompany.Text);
                cmd.Parameters.AddWithValue("@OrgNumber",
txtOrganisationNumber.Text);
                cmd.Parameters.AddWithValue("@Contact", txtContact.Text);
                cmd.Parameters.AddWithValue("@Attention", txtAttention.Text);
                cmd.Parameters.AddWithValue("@Address", txtAddress.Text);
                cmd.Parameters.AddWithValue("@PostalCode", txtPostalCode.Text);
                cmd.Parameters.AddWithValue("@City", txtCity.Text);
                cmd.Parameters.AddWithValue("@Country", txtCountry.Text);

                // Open the connection
                cn.Open();

                // Execute the INSERT statement and get the Identity number.
                Identity = Convert.ToInt64(cmd.ExecuteScalar());
            }
        }
    }
    catch (SqlException SqlEx)
    {
        Response.Write(SqlEx.Message);
    }
    catch (Exception ex)
    {
        Response.Write(ex.Message);
    }

    // When we have inserted an order we want to insert the product rows for the
order in the "OrdersProducts" table.
    // We first check to see that the identity variable not has a blank value
before we insert data to the
    // "OrdersProducts" table.
    if (Identity > 0)
    {
        // We iterate through each repeater row in the "CartRepeater" and insert
every row in the
        // "OrdersProduct" table.

```

```

        foreach (RepeaterItem RepeaterRow in CartRepeater.Items)
        {
            Literal ProductIDObj =
(Literal)RepeaterRow.FindControl("ltProductID");
            //Literal VATObj = (Literal)RepeaterRow.FindControl("ltVAT");
            TextBox QuantityObj = (TextBox)RepeaterRow.FindControl("txtQuantity");
            Literal PriceObj = (Literal)RepeaterRow.FindControl("ltPrice");

            try
            {
                string ConnString =
ConfigurationManager.ConnectionStrings["ConnectionString"].ToString();
                string sql = "INSERT INTO OrdersProducts (OrderID, ProductID,
Quantity, PriceExSaleTax) VALUES (@OrderID, @ProductID, @Quantity, @PriceExSaleTax)";

                // The using block is used to call dispose (close) automatically
even if there are an exception.
                using (SqlConnection cn = new SqlConnection(ConnString))
                {
                    using (SqlCommand cmd = new SqlCommand(sql, cn))
                    {
                        cmd.Parameters.AddWithValue("@OrderID", Identity);
                        cmd.Parameters.AddWithValue("@ProductID",
ProductIDObj.Text);

                        cmd.Parameters.AddWithValue("@Quantity",
Convert.ToDecimal(QuantityObj.Text));
                        cmd.Parameters.AddWithValue("@PriceExSaleTax",
PriceObj.Text);

                        cn.Open();
                        cmd.ExecuteNonQuery();
                    }
                }
            }
            catch (SqlException Sqlex)
            {
                Response.Write(Sqlex.Message);
            }
            catch (Exception ex)
            {
                Response.Write(ex.Message);
            }
        }

        // Delete the HttpCookie, we declare the variable CartCookie to check if
the HttpCookie "CartCookie"
        // exists and to be able to remove it.
        HttpCookie CartCookie = Request.Cookies.Get("CartCookie");

        // We check if the CartCookie exists and then sets the expiration date to
current DateTime minus one hour
        if (CartCookie != null)
        {
            CartCookie.Expires = DateTime.Now.AddHours(-1);
            Response.Cookies.Add(CartCookie);
        }

        // Redirect the user to the order list webpage
        Response.Redirect("Orders.aspx");

```

```

    }
}

protected void CalculateCartSums()
{
    // We declare variables to calculate totals for the shopping cart.

    decimal PriceTotal = 0m;
    decimal VatTotal = 0m;

    // We iterate through each row in the "CartRepeater" and add values to our
    three variables
    foreach (RepeaterItem RepeaterRow in CartRepeater.Items)
    {
        //Literal VatObj = (Literal)RepeaterRow.FindControl("ltVAT");
        Literal PriceExVat = (Literal)RepeaterRow.FindControl("ltPrice");
        TextBox QuantityObj = (TextBox)RepeaterRow.FindControl("txtQuantity");

        // Additions for totals
        //PriceTotal += Convert.ToDecimal(PriceExVat.Text) *
        Convert.ToDecimal(QuantityObj.Text);
        //VatTotal += Convert.ToDecimal(PriceExVat.Text) *
        Convert.ToDecimal(QuantityObj.Text) * (Convert.ToDecimal(VatObj.Text) / 100);

    }

    // We set the totals to labels under our repeater control and calculate the
    totalsum
    lblPriceTotal.Text = Convert.ToString(PriceTotal);
    //lblVatTotal.Text = Convert.ToString(VatTotal);
    //lblTotalSum.Text = Convert.ToString(PriceTotal + VatTotal);
}
}

```

## CAPTCHAPAGE

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Drawing.Imaging;

public partial class Default3 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        CaptchaImage ci = new
        CaptchaImage(this.Session["CaptchaImageText"].ToString(), 200, 50, "Arial Black");

        // Change the response headers to output a JPEG image.
        this.Response.Clear();
        this.Response.ContentType = "image/jpeg";
    }
}

```

```

        // Write the image to the response stream in JPEG format.
        ci.Image.Save(this.Response.OutputStream, ImageFormat.Jpeg);

        // Dispose of the CAPTCHA image object.
        ci.Dispose();
    }
}

```

## FEEDBACK PAGE

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;
using System.Net.Mail;

public partial class Default2 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        SqlConnection con = new
SqlConnection(ConfigurationManager.ConnectionStrings["ConnectionString"].ToString());
        SqlCommand cmd = new SqlCommand();
        con.Open();
        cmd.Connection = con;
        cmd.CommandText = "insert into feedback ( username, generalfeedback,
reportbug, idea, email, phone ) values ( @u,@g, @r, @i, @e, @p)";
        cmd.Parameters.AddWithValue("@g", TextBox1.Text);
        cmd.Parameters.AddWithValue("@r", TextBox2.Text);
        cmd.Parameters.AddWithValue("@i", TextBox3.Text);
        cmd.Parameters.AddWithValue("@e", TextBox4.Text);
        cmd.Parameters.AddWithValue("@p", TextBox5.Text);
        cmd.Parameters.AddWithValue("@u", "SK" + TextBox5.Text + createpassword(3));

        cmd.ExecuteNonQuery();
        Response.Write("Data Inserted");

        using (SqlConnection con1 = new
SqlConnection(ConfigurationManager.ConnectionStrings["abc"].ToString()))
        {
            SqlCommand cmd1 = new SqlCommand();
            con1.Open();
            cmd1.Connection = con1;
            cmd1.CommandText = "select * from feedback where email=@em";
            cmd1.Parameters.AddWithValue("@em", TextBox4.Text);

```



```

        SqlDataReader dr;
        dr = cmd1.ExecuteReader();
        if (dr.HasRows)
        {
            dr.Read();
            Label1.Text = "your Id Is" + dr["username"].ToString();
        }
        con1.Close();
    }

    //try
    //{
        DataSet ds = new DataSet();
        using (SqlConnection con1 = new
SqlConnection(ConfigurationManager.ConnectionStrings["abc"].ToString()))
        {
            con1.Open();

            SqlCommand cmd1 = new SqlCommand();
            cmd1.Connection = con1;
            cmd1.CommandText = "select * from feedback where email=@e";
            cmd1.Parameters.AddWithValue("@e", TextBox4.Text);
            SqlDataAdapter ad = new SqlDataAdapter(cmd1);
            ad.Fill(ds);
        }
        if (ds.Tables[0].Rows.Count > 0)
        {
            MailMessage msg = new MailMessage();
            msg.From = new MailAddress("bhanujatrehan@gmail.com");
            msg.To.Add("bhanujatrehan@gmail.com");
            msg.Subject = "Feedback";

            msg.Body = "Hi <br/> Please Check The FeedBack <br/><br/> Your UserName:"
+ ds.Tables[0].Rows[0]["username"] + "<br/> <br/> Your Email" + "&nbsp;&nbsp;&nbsp;" +
ds.Tables[0].Rows[0]["email"];
            msg.IsBodyHtml = true;
            SmtpClient smtp = new SmtpClient();
            smtp.Host = "smtp.gmail.com";
            smtp.Port = 587;
            smtp.Credentials = new
System.Net.NetworkCredential("bhanujatrehan@gmail.com", "trehanbhanu");
            smtp.EnableSsl = true;
            smtp.Send(msg);
            Response.Write("Email sent ");

            con.Close();
        }
    }

    public static string createpassword(int PasswordLength)
    {
        string allowchaos = "0123456789";
        Random rndno = new Random();
        Char[] chaos = new char[PasswordLength];

```

```

        int allowedcharcount = allowchaos.Length;
        for (int i = 0; i < PasswordLength; i++)
        {
            chaos[i] = allowchaos[(int)((allowchaos.Length) * rndno.NextDouble())];
        }
        return new string(chaos);
    }
}

```

## CONTACT US PAGE

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Net.Mail;

public partial class _Default : System.Web.UI.Page
{
    private Random random = new Random();

    private void Page_Load(object sender, System.EventArgs e)
    {
        if (!this.IsPostBack)
        {
            // Create a random code and store it in the Session object.
            this.Session["CaptchaImageText"] = GenerateRandomCode();
        }
    }

    //
    // Returns a string of six random digits.
    //
    private string GenerateRandomCode()
    {
        string s = "";
        for (int i = 0; i < 6; i++)
            s = String.Concat(s, this.random.Next(10).ToString());
        return s;
    }

    protected void txtSubmit_Click(object sender, EventArgs e)
    {
        MailMessage Message = new MailMessage();
        Message.From = new MailAddress(txtEmail.Text, txtName.Text +
Page.User.Identity.Name);
        Message.To.Add(new MailAddress("contact@yourdomain.com"));
        Message.Body = txtBody.Text;
        Message.Subject = "Contact Us";
        Message.IsBodyHtml = true;
        try
        {
            if (this.CodeNumberTextBox.Text ==
this.Session["CaptchaImageText"].ToString())
            {
                this.MessageLabel.Text = "";
            }
        }
    }
}

```

```

        SmtpClient mailClient = new SmtpClient();
        mailClient.Send(Message);
        lblFeedbackOK.Visible = true;
    }
    else
    {
        // Display an error message.
        this.MessageLabel.Text = "ERROR: Incorrect, try again.";
        // Clear the input and create a new random code.
        this.CodeNumberTextBox.Text = "";
        lblFeedbackOK.Visible = false;
        this.Session["CaptchaImageText"] = GenerateRandomCode();
    }
}
catch (Exception ex)
{
    lblFeedbackKO.Visible = true;
}
}
}

```

## 7. TESTING

Testing is the process of executing a system with the intent of finding an error. It is defined as the process in which defects are identified, isolated, subjected for rectification and ensured that product is defect free in order to produce the quality product and hence customer satisfaction. The quality is defined as justification of the requirements. Defect is nothing but deviation from the requirements and bug. Testing can demonstrate the presence of bugs, but not their absence. Debugging and Testing is not the same thing. Testing is a systematic attempt to break a program. Debugging is the art or method of uncovering why the script /program did not execute properly.

### Testing Methodologies:

- Black box (Functional) Testing: is the testing process in which tester can perform testing on an application without having any internal structural knowledge of application. Usually Test Engineers are involved in the black box testing.
- White box (Structural) Testing: is the testing process in which tester can perform testing on an application with having internal structural knowledge. Usually The Developers are involved in white box testing.

- Gray Box Testing: is the process in which the combination of black box and white box tonics' are used.

## **7.1 Functional Testing**

This testing method considers a module as a single unit and checks the unit at interface and communication with other modules rather getting in details at statement level. Here the module will be treated as a block that will take some input and generate output. Output for a given set of input combinations are forwarded to other modules.

Black-box test are designed to uncover errors functional requirement without regard to the internal workings of a program. Black-box testing techniques focus on the information domain of the software, deriving test cases by partitioning the input and output domain of a program in manner that provides through test coverage. The black-box test is used to demonstrate that software functions are operational, that input is properly produced, and the integrity of external information is maintained. The black-box test examines some fundamental aspect of a system with little or no regard for the logical structure of the software.

Graph based testing methods explore the relationship between and behaviour of program objects. Equivalence partitioning divides the input classes of data are likely to exercise specific software function. Boundary values analysis probes the program's ability to handle data at the limits of acceptability.

## **7.2 Structural Testing**

This is a unit testing method where a unit will be taken at a time and tested thoroughly at a statement level to find the maximum possible errors. The white box testing is called Glass Box Testing.

We tested step wise every piece of code, taking care that every statement in the code is executed at least once. We have generated a list of test cases, sample data, which is used to check all possible combinations of execution paths through the code at every module level.

White-box test focuses on the program control structure. Test cases are derived to ensure that all statement in the program control structure has been executed at least once during testing and that all logical conditions have been exercised. Basis path testing, a white box technique,

makes use of program graphs (or graph matrices) to derive the set of linearly independent test that will ensure coverage. Condition and data flow testing further exercising degrees of complexity.

According to the need of the software, the following testing plans have been planned on some amount on test data. Hypothetical data is used to test the system before implementation. Some temporary user ids are created to check the validity and authenticity of the users. Various constraints are checked for their working. A demo case will be taken with dummy data for new users.

### **7.3 Levels Of Testing**

#### **7.3.1 System Testing**

Testing is a set activity that can be planned and conducted systematically. Testing begins at the module level and work towards the integration of entire computers based system. Nothing is complete without testing, as it is vital success of the system.

- **Testing Objectives:**

There are several rules that can serve as testing objectives, they are

1. Testing is a process of executing a program with the intent of finding an error
2. A good test case is one that has high probability of finding an undiscovered error.
3. A successful test is one that uncovers an undiscovered error.

If testing is conducted successfully according to the objectives as stated above, it would uncover errors in the software. Also testing demonstrates that software functions appear to the working according to the specification, that performance requirements appear to have been met.

There are three ways to test a program

1. For Correctness
2. For Implementation efficiency
3. For Computational Complexity.

Tests for correctness are supposed to verify that a program does exactly what it was designed to do. This is much more difficult than it may at first appear, especially for large programs.

Tests for implementation efficiency attempt to find ways to make a correct program faster or use less storage. It is a code-refining process, which re-examines the implementation phase of algorithm development. Tests for computational complexity amount to an experimental

analysis of the complexity of an algorithm or an experimental comparison of two or more algorithms, which solve the same problem.

- **Testing Correctness**

The following ideas should be a part of any testing plan:

1. Preventive Measures
2. Spot checks
3. Testing all parts of the program
4. Test Data
5. Looking for trouble
6. Time for testing
7. Re Testing

The data is entered in all forms separately and whenever an error occurred, it is corrected immediately. A quality team deputed by the management verified all the necessary documents and tested the Software while entering the data at all levels

### **7.3.2 Unit Testing**

As this system was partially GUI based WINDOWS application, the following were tested in this phase

1. Tab Order
2. Reverse Tab Order
3. Field length
4. Front end validations

In our system, Unit testing has been successfully handled. The test data was given to each and every module in all respects and got the desired output. Each module has been tested found working properly.

### **7.3.3 Integration Testing**

Test data should be prepared carefully since the data only determines the efficiency and accuracy of the system. Artificial data are prepared solely for testing. Every program validates the input data.

### **7.3.4 Validation Testing**

In this, all the Code Modules were tested individually one after the other. The following were tested in all the modules

1. Loop testing
2. Boundary Value analysis
3. Equivalence Partitioning Testing

In our case all the modules were combined and given the test data. The combined module works successfully without any side effect on other programs. Everything was found fine working.

### **7.3.5 Output Testing**

This is the final step in testing. In this the entire system was tested as a whole with all forms, code, modules and class modules. This form of testing is popularly known as Black Box testing or system testing.

Black Box testing methods focus on the functional requirement of the software. That is, Black Box testing enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements for a program. Black Box testing attempts to find errors in the following categories; incorrect or missing functions, interface errors, errors in data structures or external database access, performance errors and initialization errors and termination errors.

## **8. IMPLEMENTATION**

### **8.1 Implementation of the Project**

#### **ASP.NET-Front End of Website**

ASP.NET makes building real world Web applications dramatically easier. ASP.NET server controls enable an HTML-like style of declarative programming that let you build great pages with far less code than with classic ASP. Displaying data, validating user input, and uploading files are all amazingly easy. Best of all, ASP.NET pages work in all browsers including Netscape, Opera, ..ASP.NET lets you leverage your current programming language skills. Unlike classic ASP, which supports only interpreted VBScript and J Script, ASP.NET

now supports more than 25 .NET languages (built-in support for VB.NET, C#, and JScript.NET), giving us unprecedented flexibility in the choice of language.

#### SQL SERVER EXPRESS 2008-Back End of Website

Microsoft SQL Server 2008 Management Studio Express is a free, integrated environment for accessing, configuring, managing, administering, and developing all components of SQL Server, as well as combining a broad group of graphical tools and rich script editors that provide access to SQL Server to developers and administrators of all skill levels. SQL Server 2008 was released on August 6, 2008 and aims to make data management self-tuning, self-organizing, and self-maintaining with the development of *SQL Server Always On* technologies, to provide near-zero downtime. SQL Server 2008 also includes support for structured and semi-structured data, including digital media formats for pictures, audio, video and other multimedia data.

### 8.2 Conversion Plan (How Website Works)

All the way like the working of the other websites our site also work like that once the user writes the website name (URL) in the web browser it converts that typed name into the particular IP if it exists for that particular website and sends the request to the server.

Now what actually happens the browser contacts the server and requests that the server deliver the document to it. The server then gives a response which contains the document and the browser displays this to the user. The server also tells the browser what kind of document it is (HTML file, PDF file, ZIP file etc) and the browser then shows the document with the program it was configured to use for this kind of document. The browser will display HTML documents directly, and if there are references to images, JSP, sound clips etc in it and the browser has been set up to display these it will request these also from the servers on which they resides. It's worth noting that these will be separate requests, and add additional load to the server and network. When the user follows another link the whole sequence starts a new processing. These requests and responses are issued in a special language called HTTP, which is short for Hyper Text Transfer Protocol. HTTP only defines what the browser and web server say to each other, not how they communicate. The actual work of moving bits and bytes back and forth across the network is done by TCP and IP. In our website the technology what is used is explained above now how the code is working and how the data flow between the server and client (browser) and user takes place.



### 8.3 Post implementation and Software Maintenance:

First of all after the user fills the URL for online gaming portal home page is displayed which asks the user whether he/she is a registered user or not if the user is already is a member so he/she must go to login page where they can authenticate to website.

#### Software Maintenance

Software maintenance in software engineering is the modification of a software product after delivery to correct faults, to improve performance or other attributes.

Once the software is delivered and deployed, then maintenance phase starts. Software requires maintenance because there are some residual errors remaining in the system that must be removed as they are discovered. Maintenance involves understanding the existing software (code and related documents), understanding the effect of change, making the changes, testing the new changes, and retesting the old parts that were not changed. The complexity of maintenance task makes maintenance the most costly activity in the life of software product.

The keys to reduce the need for maintenance are:-

- More accurately defining the user's requirement during system development.
- Preparation of system documentation in better way.
- Using more effective ways for designing processing logic and communicating it to project team members.
- Making better use of exiting tools and techniques.
- Software must be maintained when:
- The reality the software models changes, New functionality is added, It is easier to change software than hardware, Software must be updated to run on improved hardware or with improved software.

There are several types of software maintenance. They are:

Maintenance Type	Description
Corrective	Fixes a fault in the software without changing or adding to the software's functionality.
Adaptive	Modifies software to preserve functionality in a changed environment.
Perfective	Improves software performance, maintainability, etc., and can extend the functionality of the application.
Preventive	Changes are made to the system in order to prevent further faults and to

	improve the structure and maintainability of the system.
--	--

## **9. Project legacy**

### **9.1 Current status of the project**

Website contain four modules homepage, product page, feedback and contact us. Homepage contains basic description such as login page, sign up and link to other pages. Product page include all the product details, online order for all the products. Feedback page provide the facility for the customers for online assistance and any suggestion or feedback made sends email to the admin page. Contact us customers can contact the company anytime for their help or for placing order. Except it all the modules of our website is completed and any changes in the source if necessary is required that are done as per the future needs.

### **9.2 Remaining areas of concern**

Our Site includes all the features /including user interactivity with the site but lacks in the online payment. In blocs news updation can be added according to the company requirement in near future.

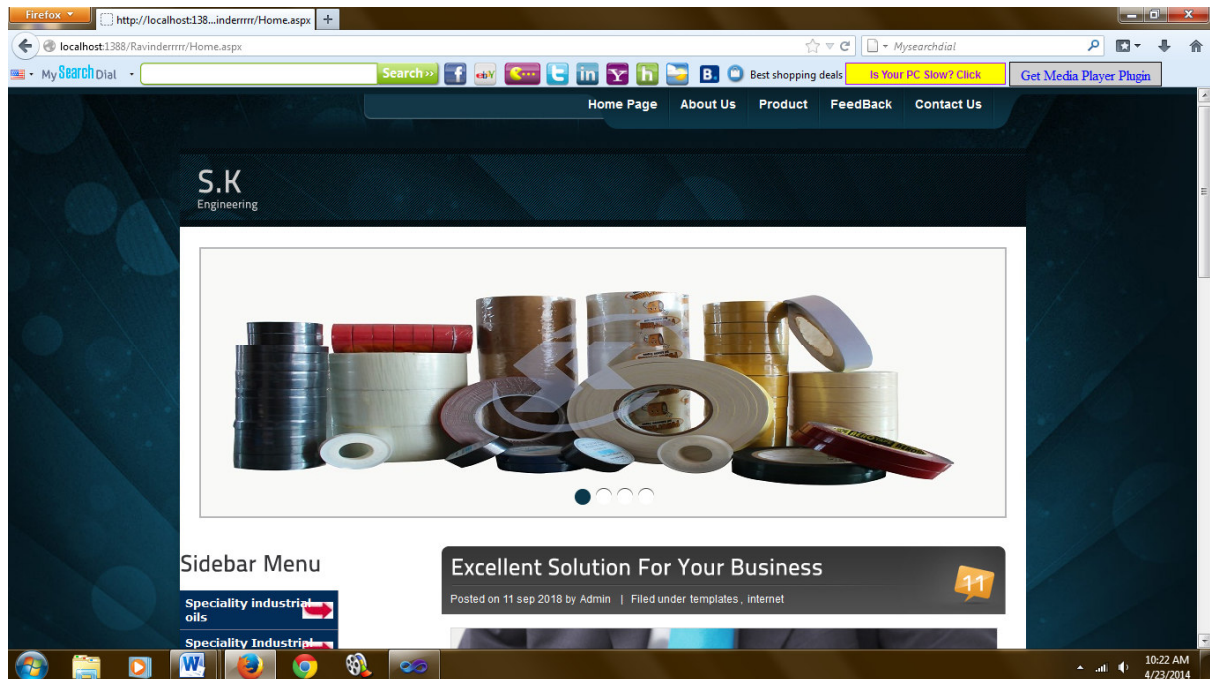
### **9.3 Technical and Managerial lesson learnt**

Technical and Managerial lesson learnt through my project implementation live website for a company Automation Of business I come to learn many languages some of which are implemented in the project including ASP AND SQL SERVER 2008.

Regarding the managerial lesson we learnt the team work, patience, cooperation, coordination.

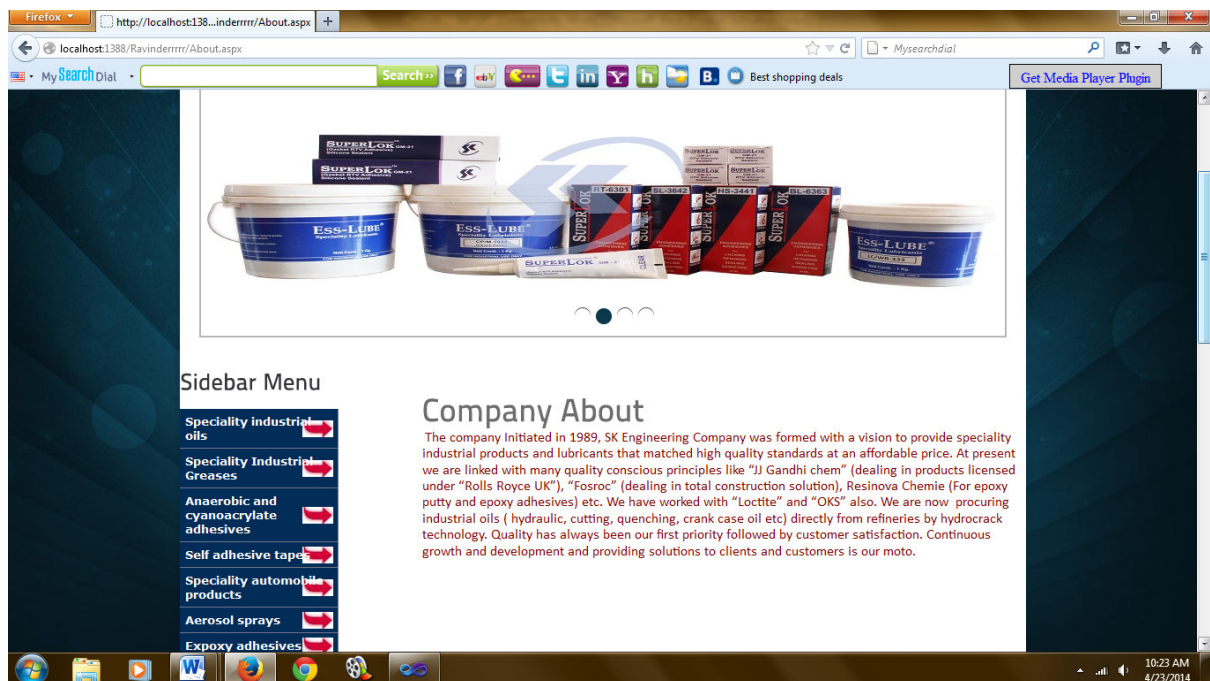
## 10. USER MANUAL

When a user will enter the URL for website the home page will appear



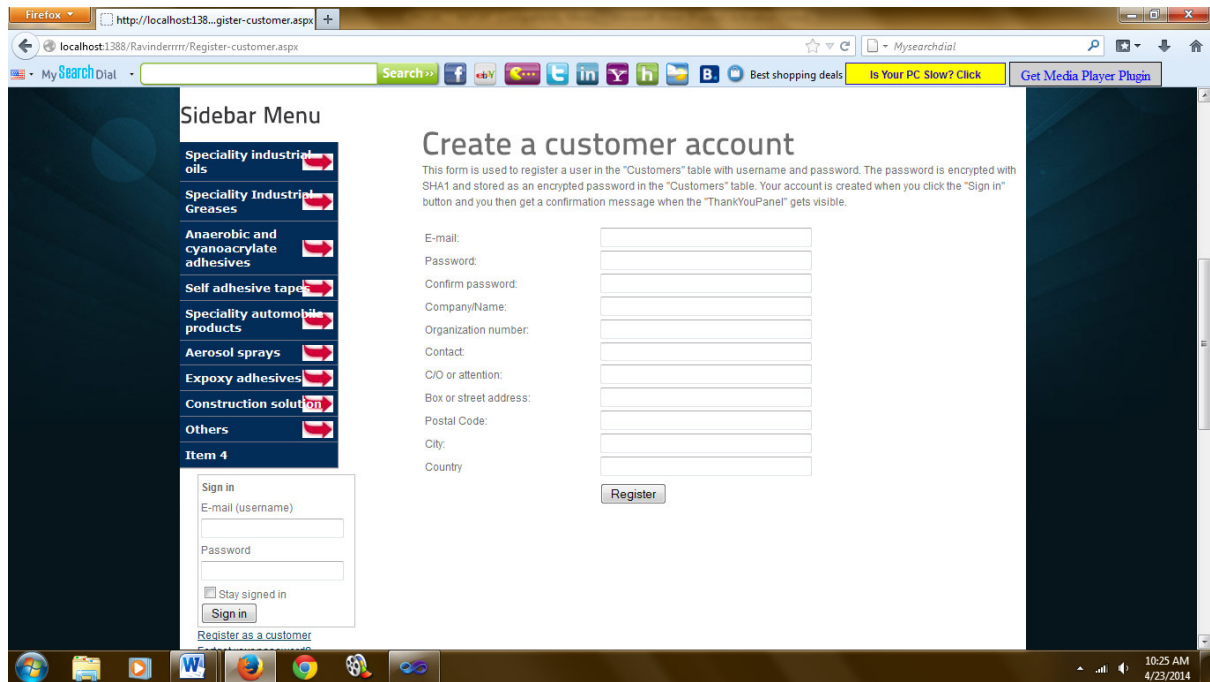
Snapshot : 1

If the user want to know about the company then the user can view the content on the About Us page where the details of the company are available.



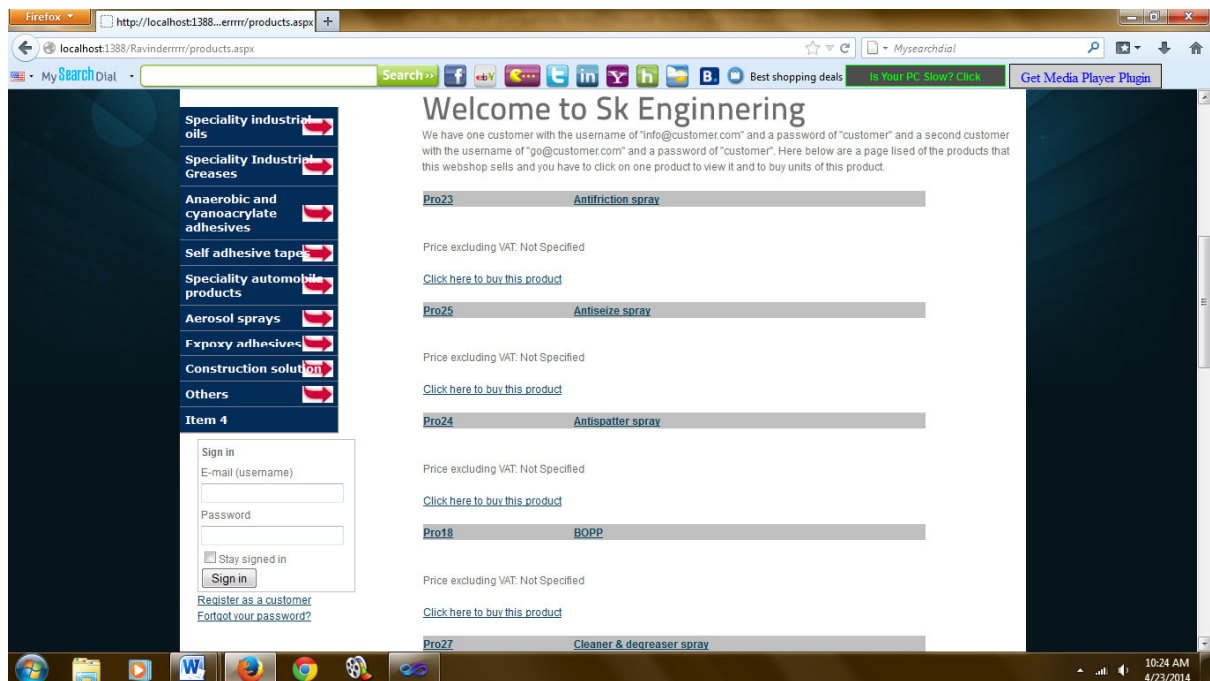
Snapshot : 2

The customer has to create an account if he wants to purchase the products from the company. i.e. user has to Register himself/herself for purchasing Products from the Website.



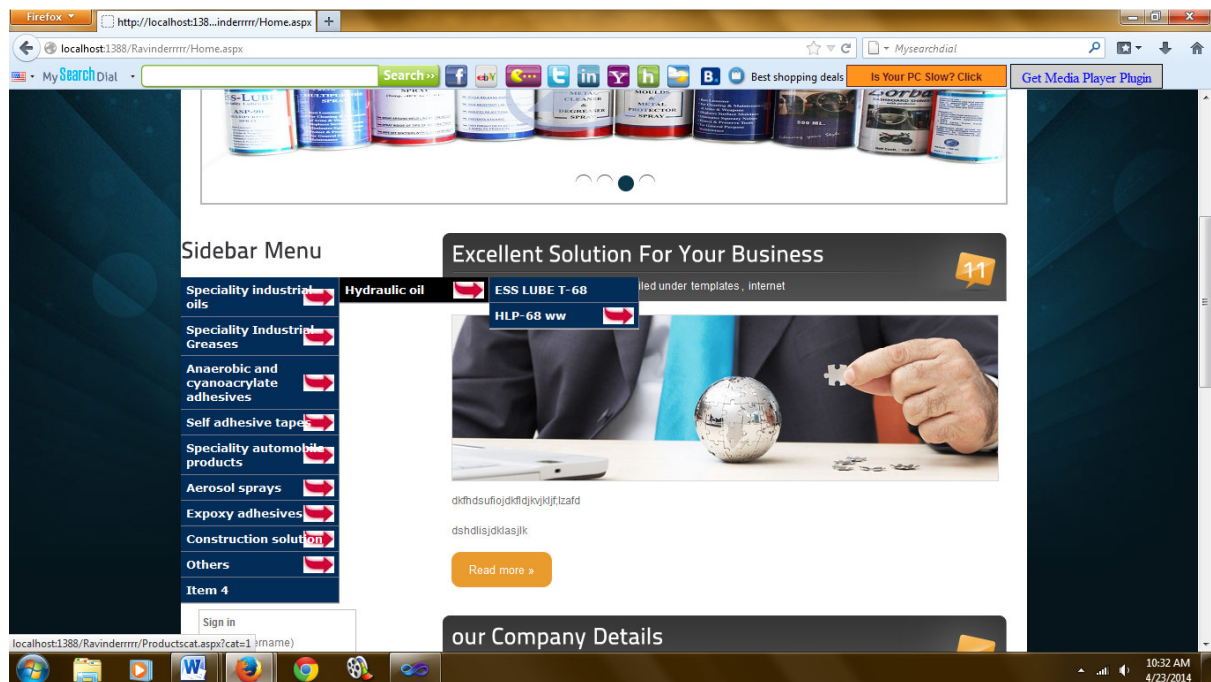
Snapshot : 3

The details of the products can be seen on the product page and the user can place an order by selecting the products from the product page only if he/she has already Register otherwise he/she has to Register first.



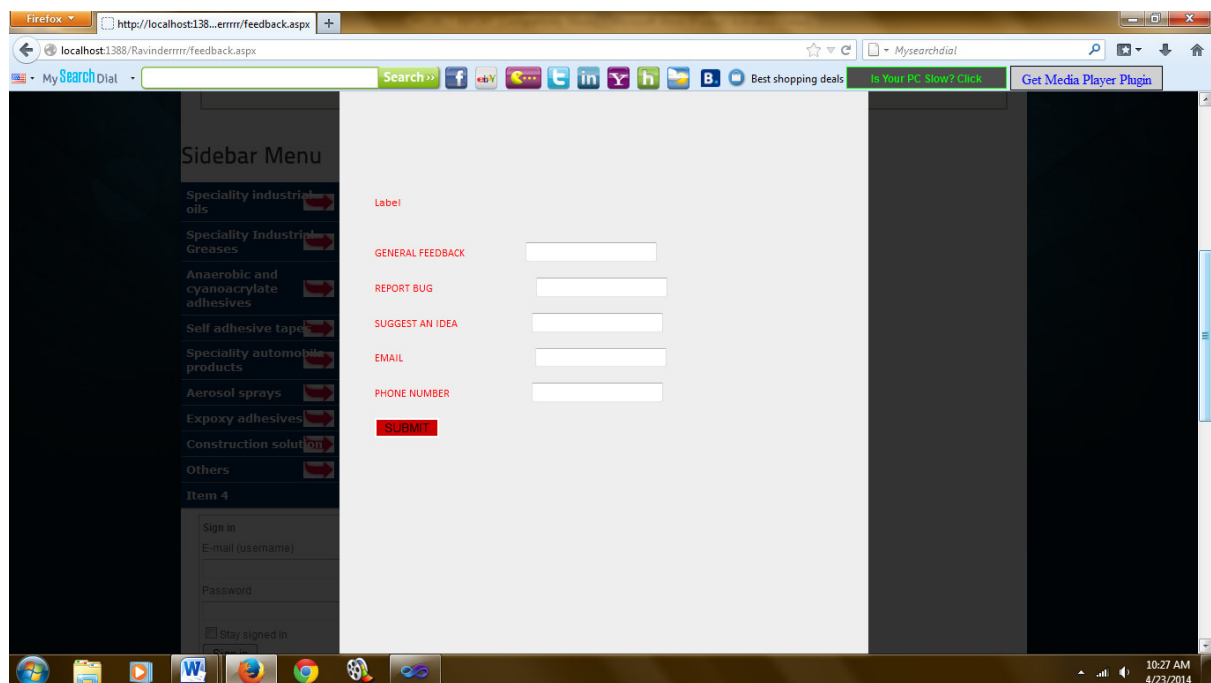
Snapshot : 4

Side bar menu for the viewing the products is also available having a drop down list for the product to search the product under particular category.



Snapshot : 5

The registered customer can give the feedback on the feedback page and this will be sent to the owner e-mail from where he can check the details.



Snapshot : 6

If the customer want that owner will contact that particular person then the customer can give his contact information on the contact us page.





```

        <div class="clr"></div>
        <div class="img"></div>
        <div class="post_content">
            <p>asdasdsadas</p>
            <p><strong>dsasad</strong>adadasdasdas</p>
            <p class="spec"><a href="#" class="rm">Read more &raquo;</a></p>
        </div>
        <div class="clr"></div>
    </div>
    <p class="pages"><small>Page 1 of 2</small> <span>1</span> <a href="#">2</a>
<a href="#">&raquo;</a></p>
</asp:Content>

```

## PRODUCT PAGE

```

<%@ Page Title="" Language="C#" MasterPageFile="~/MasterPage2.master"
AutoEventWireup="true" CodeFile="Product.aspx.cs" Inherits="Default2" %>

```

```

<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
<h1><asp:Literal ID="ltProductName" runat="server"></asp:Literal></h1>
<asp:Literal ID="ltDescription" runat="server"></asp:Literal><br /><br />
Price excluding VAT: <asp:Literal ID="ltPriceExSaleTax" runat="server"
></asp:Literal><br />
Sale tax (VAT %): <asp:Literal ID="ltSaleTax" runat="server"></asp:Literal>
(<asp:Literal ID="ltSaleTaxPercent" runat="server"></asp:Literal>)<br />
Price including VAT: <asp:Literal ID="ltTotalPrice" runat="server"></asp:Literal><br
/><br />
Quantity: <asp:TextBox ID="txtQuantity" Width="50px" runat="server"
Text="1"></asp:TextBox> <asp:Button ID="btnBuy" runat="server" Text="Buy"
OnClick="btnBuy_Click" /><br /><br />
<asp:HiddenField ID="HiddenProductID" runat="server" />
</asp:Content>

```

```

<%@ Page Title="" Language="C#" MasterPageFile="~/MasterPage2.master"
AutoEventWireup="true" CodeFile="products.aspx.cs" Inherits="Default2" %>

```

```

<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
    <div class="extra">
        <h1>Welcome to Sk Enginnering</h1>
        We have one customer with the username of "info@customer.com" and a password
of "customer" and a second customer
        with the username of "go@customer.com" and a password of "customer". Here
below are
        a page listed of the products that this webshop sells and you have to click on
one product to view it and to
        buy units of this product.<br /><br />
        <asp:DataList ID="ProductListRepeater" runat="server" RepeatColumns="3"
RepeatDirection="Horizontal">
            <HeaderTemplate>
                <table border="0" width="570px" cellpadding="0" cellspacing="0">

```

```

        <itemtemplate>
            <tr>
                <td style="width:30%; font-weight:bolder; background-
color:Silver"><asp:HyperLink ID="ltProductID" runat="server"
Text='<#DataBinder.Eval(Container.DataItem, "ProductID")%>'
NavigateUrl='<#GenerateURL(Eval("ProductID"))%>'></asp:HyperLink></td>
                <td style="width:70%; font-weight:bolder; background-
color:Silver"><asp:HyperLink ID="ltProductName" runat="server"
Text='<#DataBinder.Eval(Container.DataItem, "ProductName")%>'
NavigateUrl='<#GenerateURL(Eval("ProductID"))%>'></asp:HyperLink></td>
            </tr>
            <tr>
                <td colspan="2">
                    <asp:Literal ID="ltDescription" runat="server"
Text='<#DataBinder.Eval(Container.DataItem, "Description")%>'></asp:Literal><br /><br />
                    Price excluding VAT: <asp:Literal ID="ltPriceExSaleTax"
runat="server" Text='<#DataBinder.Eval(Container.DataItem,
"PriceExSaleTax")%>'></asp:Literal><br />
                    <%-- Sale tax (VAT %): <asp:Literal ID="ltSaleTax"
runat="server" Text='<#DataBinder.Eval(Container.DataItem,
"SaleTaxMoney")%>'></asp:Literal> (<asp:Literal ID="ltSaleTaxPercent" runat="server"
Text='<#DataBinder.Eval(Container.DataItem, "SaleTaxPercent",
"{0:P}"%>'></asp:Literal><br />
                    Price including VAT: <asp:Literal ID="ltTotalPrice"
runat="server" Text='<#DataBinder.Eval(Container.DataItem,
"TotalPrice")%>'></asp:Literal><br /><--%><br />
                    <asp:HyperLink ID="hplLinkToProduct" runat="server"
NavigateUrl='<#GenerateURL(Eval("ProductID"))%>'>Click here to buy this
product</asp:HyperLink><br /><br />
                </td>
            </tr>
        </itemtemplate>
    </FooterTemplate>
</table>
</FooterTemplate>
</asp:DataList>
    <asp:HyperLink id="lnkPrev" Visible="false" runat="server"
Text="Previous"></asp:HyperLink>
    <asp:Label id="lblCurrentPage" runat="server" />
    <asp:HyperLink id="lnkNext" Visible="false" runat="server"
Text="Next"></asp:HyperLink><br /><br />
</div>
</asp:Content>

```

## PRODUCT CART PAGE

```

<%@ Page Title="" Language="C#" MasterPageFile="~/MasterPage2.master"
AutoEventWireup="true" CodeFile="Productscat.aspx.cs" Inherits="Default2" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
    <asp:DataList ID="DataList1" runat="server" CellPadding="4"
        DataKeyField="ProductID" DataSourceID="SqlDataSource1" ForeColor="#333333"
        Width="470px">
        <AlternatingItemStyle BackColor="White" ForeColor="#284775" />
        <FooterStyle BackColor="#5D7B9D" Font-Bold="True" ForeColor="White" />
    </asp:DataList>

```



```

<HeaderStyle BackColor="#5D7B9D" Font-Bold="True" ForeColor="White" />
<ItemStyle BackColor="#F7F6F3" ForeColor="#333333" />
<ItemTemplate>
    ProductID:
    <asp:Label ID="ProductIDLabel" runat="server" Text='<%# Eval("ProductID")
%>' />

    <br />
    ProductName:
    <asp:Label ID="ProductNameLabel" runat="server"
        Text='<%# Eval("ProductName") %>' />
    <br />
    PriceExSaleTax:
    <asp:Label ID="PriceExSaleTaxLabel" runat="server"
        Text='<%# Eval("PriceExSaleTax") %>' />
    <br />
<br />
<asp:HyperLink ID="hp1LinkToProduct" runat="server"
NavigateUrl='<%#GenerateURL(Eval("ProductID"))%>'>Click here to buy this
product</asp:HyperLink><br /><br />
</ItemTemplate>
    <SelectedItemStyle BackColor="#E2DED6" Font-Bold="True" ForeColor="#333333" />
</asp:DataList>
<%--<asp:DetailsView ID="DetailsView1" runat="server" Height="184px"
Width="461px" AutoGenerateRows="False" CellPadding="4" DataKeyNames="ProductID"
DataSourceID="SqlDataSource1" ForeColor="#333333" GridLines="None">
    <AlternatingRowStyle BackColor="White" ForeColor="#284775" />
    <CommandRowStyle BackColor="#E2DED6" Font-Bold="True" />
    <EditRowStyle BackColor="#999999" />
    <FieldHeaderStyle BackColor="#E9ECF1" Font-Bold="True" />
    <Fields>
        <asp:BoundField DataField="ProductID" HeaderText="ProductID"
ReadOnly="True"
            SortExpression="ProductID" />
        <asp:BoundField DataField="ProductName" HeaderText="ProductName"
            SortExpression="ProductName" />
        <asp:BoundField DataField="PriceExSaleTax" HeaderText="PriceExSaleTax"
            SortExpression="PriceExSaleTax" />
    </Fields>
    <FooterStyle BackColor="#5D7B9D" Font-Bold="True" ForeColor="White" />
    <HeaderStyle BackColor="#5D7B9D" Font-Bold="True" ForeColor="White" />
    <PagerStyle BackColor="#284775" ForeColor="White" HorizontalAlign="Center" />
    <RowStyle BackColor="#F7F6F3" ForeColor="#333333" />
</asp:DetailsView>--%>

<asp:SqlDataSource ID="SqlDataSource1" runat="server"
    ConnectionString='<%= $ ConnectionStrings:ConnectionString %>'
    SelectCommand="SELECT [ProductID], [ProductName], [PriceExSaleTax] FROM [Products]
WHERE ([cat] = @cat)">
    <SelectParameters>
        <asp:QueryStringParameter DefaultValue="string" Name="cat"
            QueryStringField="cat" Type="String" />
    </SelectParameters>
</asp:SqlDataSource>

</asp:Content>

```

## REGISTRATION PAGE

```

<%@ Page Title="" Language="C#" MasterPageFile="~/MasterPage2.master"
AutoEventWireup="true" CodeFile="Register-customer.aspx.cs" Inherits="Default2" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
<asp:Panel ID="CurrentPanel" runat="server" Visible="true">
<h1>Create a customer account</h1>
This form is used to register a user in the "Customers" table with username and
password. The password is encrypted with
SHA1 and stored as an encrypted password in the "Customers" table. Your account is
created when you click the
"Sign in" button and you then get a confirmation message when the "ThankYouPanel" gets
visible.<br />
<br />
<table width="704">
    <tr>
        <td style="width: 200px">
            E-mail:</td>
        <td style="width: 254px">
            <asp:TextBox id="txtUserName" runat="server"
Width="95%"></asp:TextBox></td>
        <td style="width: 250px">
            <asp:RequiredFieldValidator id="EmailRequired" runat="server"
ControlToValidate="txtUserName"
                Display="Dynamic" ErrorMessage="* A valid email must be entered"
Width="216px" ValidationGroup="RegisterCustomer">* A valid email must be
entered</asp:RequiredFieldValidator></td>
    </tr>
    <tr>
        <td style="width: 200px">Password:</td>
        <td style="width: 254px">
            <asp:TextBox id="txtPassword" textmode="Password" runat="server"
width="95%"></asp:TextBox></td>
        <td style="width: 250px">
            <asp:RequiredFieldValidator id="PasswordRequired" runat="server"
controltovalidate="txtPassword"
                ErrorMessage="* Password must be entered"
validationgroup="RegisterCustomer" display="Dynamic" width="216px">* Password must be
entered</asp:RequiredFieldValidator></td>
    </tr>
    <tr>
        <td style="width: 200px">Confirm password:</td>
        <td style="width: 254px">
            <asp:TextBox ID="txtConfirmPassword" textmode="Password" runat="server"
width="95%"></asp:TextBox></td>
        <td style="width: 250px">
            <asp:RequiredFieldValidator id="PasswordConfirmRequired" runat="server"
controltovalidate="txtConfirmPassword"
                ErrorMessage="* Password must be entered"
validationgroup="RegisterCustomer" display="Dynamic" width="216px">* Password must be
entered</asp:RequiredFieldValidator>
            <asp:CompareValidator id="ComparePassword" runat="server"
controltocompare="txtPassword"
                controltovalidate="txtConfirmPassword" ErrorMessage="* Passwords do
not match"
                width="216px"
validationgroup="RegisterCustomer"></asp:CompareValidator></td>
    </tr>
    <tr>
        <td style="width: 200px">Company/Name:</td>

```

```

        <td style="width: 254px">
            <asp:TextBox id="txtCompanyName" runat="server"
width="95%"></asp:TextBox></td>
        <td style="width: 250px">
            <asp:RequiredFieldValidator ID="CompanyRequired" runat="server"
controltovalidate="txtCompanyName"
            display="Dynamic" errorMessage="* Company / Name must be entered"
width="216px" validationgroup="RegisterCustomer">* Company / Name must be
entered</asp:RequiredFieldValidator></td>
        </tr>
        <tr>
            <td style="width: 200px">Organization number:</td>
            <td style="width: 254px">
                <asp:TextBox id="txtOrganisationNumber" runat="server"
width="95%"></asp:TextBox></td>
            <td style="width: 250px">
            </td>
        </tr>
        <tr>
            <td style="width: 200px">Contact:</td>
            <td style="width: 254px">
                <asp:TextBox id="txtContact" runat="server"
width="95%"></asp:TextBox></td>
            <td style="width: 250px">
            </td>
        </tr>
        <tr>
            <td style="width: 200px">
                C/O or attention:</td>
            <td style="width: 254px">
                <asp:TextBox id="txtAttention" runat="server"
width="95%"></asp:TextBox></td>
            <td style="width: 250px">
            </td>
        </tr>
        <tr>
            <td style="width: 200px">
                Box or street address:</td>
            <td style="width: 254px">
                <asp:TextBox ID="txtAdress" runat="server" width="95%"></asp:TextBox></td>
            <td style="width: 250px">
            </td>
        </tr>
        <tr>
            <td style="width: 200px">Postal Code:</td>
            <td style="width: 254px">
                <asp:TextBox id="txtPostalCode" runat="server"
width="95%"></asp:TextBox></td>
            <td style="width: 250px">
            </td>
        </tr>
        <tr>
            <td style="width: 200px">City:</td>
            <td style="width: 254px">
                <asp:TextBox id="txtCity" runat="server" width="95%"></asp:TextBox></td>
            <td style="width: 250px">
            </td>
        </tr>
        <tr>
            <td style="width: 200px">Country</td>
            <td style="width: 254px">

```

```

        <asp:TextBox id="txtCountry" runat="server"
width="95%"></asp:TextBox></td>
        <td style="width: 250px">
        </td>
    </tr>
    <tr>
        <td style="width: 200px">
        </td>
        <td style="width: 254px">
        </td>
        <td style="width: 250px">
        </td>
    </tr>
    <tr>
        <td style="width: 200px">
        </td>
        <td style="width: 254px">
            <asp:Button id="btnRegister" runat="server" text="Register"
validationgroup="RegisterCustomer" OnClick="btnRegister_Click" /></td>
        <td style="width: 250px">
            <asp:Label id="SqlEx" runat="server" forecolor="Red"
width="216px"></asp:Label></td>
    </tr>
</table>
</asp:Panel>
<asp:Panel ID="ThankYouPanel" runat="server" Visible="false">
<h1>Thank you</h1>
Thank you for register with us.<br />
<br />
</asp:Panel>
</asp:Content>

```

## SHOPPING CART PAGE

```

<%@ Page Title="" Language="C#" MasterPageFile="~/MasterPage2.master"
AutoEventWireup="true" CodeFile="Shopping-cart.aspx.cs" Inherits="Default2" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
    <h1>Shopping cart</h1>
    We use a repeater control to build our shopping cart and we fill the shopping cart
    with data from the "Products" table
    when this page loads by using data in the HttpCookie as parameters to select rows from
    the table. We have
    added EnableViewState="false" for the "Quantity" textbox in the repeater control so
    that it is possible to change
    this value in an update of the cart.<br />
    <br />
    <asp:Repeater ID="CartRepeater" runat="server"
OnItemCommand="CartRepeater_ItemCommand">
        <HeaderTemplate>
            <table border="0" width="600px" cellpadding="1" cellspacing="0" style="border:
solid 1px silver">
                <tr style="background-color:Silver">
                    <th>Product #</th>
                    <th>Product name</th>
                    <%-- <th>VAT %</th>--%>
                    <th>Quantity</th>

```

```

        <th>Product Price</th>
        <th></th>
    </tr>
</HeaderTemplate>
<ItemTemplate>
    <tr>
        <td><asp:Literal ID="ltProductID" runat="server"
Text='<%#DataBinder.Eval(Container.DataItem, "ProductID")%>'></asp:Literal></td>
        <td><asp:Literal ID="ltProductName" runat="server"
Text='<%#DataBinder.Eval(Container.DataItem, "ProductName")%>'></asp:Literal></td>
        <%--<td><asp:Literal ID="ltVAT" runat="server"
Text='<%#DataBinder.Eval(Container.DataItem, "SaletaxPercent")%>'></asp:Literal></td>-
-%>
        <td><asp:Textbox ID="txtQuantity" EnableViewState="false" Width="50px"
CssClass="textbox" runat="server" Text='<%#DataBinder.Eval(Container.DataItem,
"QuantityC")%>'></asp:Textbox></td>
        <td><asp:Literal ID="ltPrice" runat="server"
Text='<%#DataBinder.Eval(Container.DataItem, "PriceExSaleTax")%>'></asp:Literal></td>
        <td><asp:LinkButton ID="btnRemove" runat="server" Text="X"
CommandArgument='<%# Bind("ProductID") %>' /></td>
    </tr>
</ItemTemplate>

    <FooterTemplate>
    </table>
    </FooterTemplate>
</asp:Repeater>
<br />
<table border="0" width="600" cellpadding="1" cellspacing="0">
<tr>
<td style="text-align:right;">
    Sum excluding VAT: <asp:Label ID="lblPriceTotal" runat="server" Font-Bold="true"
Text=""></asp:Label><br />
    <%-- VAT in total: <asp:Label ID="lblVatTotal" runat="server" Font-Bold="true"
Text=""></asp:Label><br />--%>
    Totalsum: <asp:Label ID="lblTotalSum" runat="server" Font-Bold="true"
Text=""></asp:Label>
</td>
</tr>
</table>
<asp:Button ID="btnUpdateCart" runat="server" Text="Update cart"
OnClick="btnUpdateCart_Click" />
<asp:Button ID="btnCheckOut" runat="server" Text="Check out"
OnClick="btnCheckOut_Click" /><br /><br />
<asp:Panel ID="CheckOutPanel" runat="server" Visible="false">
<h1>Order information</h1>
    We select the order information data from the "Customers" table when the user has
    clicked the
    "Check out" button and this data can be changed. When the user clicks the "Send
    order" button
    there is code to save the order in the "Orders" table and to save the orderrows in
    the
    "OrdersProducts" table.<br /><br />
    <table border="0" width="600" cellpadding="1" cellspacing="0">
    <tr>
    <td>Company/Name:</td>
    <td><asp:TextBox ID="txtCompany" runat="server" Width="200px"></asp:TextBox></td>
    </tr>
    <tr>
    <td>ID Number:</td>
    <td><asp:TextBox id="txtOrganisationNumber" runat="server"
Width="200px"></asp:TextBox></td>

```

```

        </tr>
        <tr>
        <td>Contact:</td>
        <td><asp:TextBox id="txtContact" runat="server" Width="200px"></asp:TextBox></td>
        </tr>
        <tr>
        <td>Attention:</td>
        <td><asp:TextBox id="txtAttention" runat="server"
Width="200px"></asp:TextBox></td>
        </tr>
        <tr>
        <td>Adress:</td>
        <td><asp:TextBox ID="txtAdress" runat="server" Width="200px"></asp:TextBox></td>
        </tr>
        <tr>
        <td>Postal code:</td>
        <td><asp:TextBox id="txtPostalCode" runat="server"
Width="200px"></asp:TextBox></td>
        </tr>
        <tr>
        <td>City:</td>
        <td><asp:TextBox id="txtCity" runat="server" Width="200px"></asp:TextBox></td>
        </tr>
        <tr>
        <td>Country:</td>
        <td><asp:TextBox id="txtCountry" runat="server" Width="200px"></asp:TextBox></td>
        </tr>
    </table>
    <asp:Button ID="btnOrder" runat="server" Text="Send order"
OnClick="btnOrder_Click" /><br /><br />
    </asp:Panel>
    <asp:HiddenField ID="HiddenCustomerID" runat="server" />
    <br />
</asp:Content>

```

## 12. BIBLIOGRAPHY

Here I would like to mention the name of the books and URLs used for reference while designing, testing, implementation, and coding of the system:-

1. [www.google.co.in](http://www.google.co.in)
2. Software Engineering - A practitioner's approach



Discipline: \_\_\_\_\_

PROJECT TOPIC APPROVAL PERFORMA

GROUP NO. \_\_\_\_\_

COURSE CODE:- \_\_\_\_\_

SR.NO.	NAME OF STUDENT	REGISTRATION NO.	BATCH	SESSION	PARENT SECTION	CURRENT SECTION	ROLL NO.
1	Bhanuja Behera	11000153	2010 - 2014		K1006		01
2	Isnan Jaidka	11000236	2010 - 2014		K1006		02
3	Harpreet Kaur	11002026	2010 - 2014		K1006		09
4	Ravinder Kaur	11001113	2010 - 2014		K1006		05
5							

Details of Supervisor:

Name: Swilpa

U.ID: 13881

Designation: Lecturer

Qualification: B.Tech

Research Experience: \_\_\_\_\_

PROPOSED TOPICS

1. Live Website for Company : Automation of Business
2. Accommodation Application
3. Portal Design

Signature of Supervisor

\*Guide should finally encircle one topic out of three proposed topics and put up for approval before Project Approval Committee (PAC)

\*Original copy of this format after PAC approval will be retained by the student and must be attached in the Project/Dissertation synopsis and final report.

\*One copy to be submitted to Supervisor.

APPROVAL OF PAC CHAIRPERSON:

Signature: \_\_\_\_\_