

**THE DESIGN AND IMPLEMENTATION OF AN E-COMMERCE
SITE FOR ONLINE BOOK SALES**

By

Debajani Pradhan

Project Report

for Master In Computer Applications

In the

Gandhi Engineering College

2025

©2025

Debajani Pradhan

All Rights Reserved

Abstract

The business-to-consumer aspect of electronic commerce (e-commerce) is the most visible business use of the World Wide Web. The primary goal of an e-commerce site is to sell goods and services online.

This project deals with developing an e-commerce website for Online Book Sale. It provides the user with a catalog of different books available for purchase in the store. In order to facilitate online purchase a shopping cart is provided to the user. The system is implemented using a 3-tier approach, with a backend database, a middle tier of Microsoft Internet Information Services (IIS) and ASP.NET, and a web browser as the front end client.

In order to develop an e-commerce website, a number of Technologies must be studied and understood. These include multi-tiered architecture, server and client side scripting techniques, implementation technologies such as ASP.NET, programming language (such as C#, VB.NET), relational databases (such as MySQL, Access).

This is a project with the objective to develop a basic website where a consumer is provided with a shopping cart application and also to know about the technologies used to develop such an application.

This document will discuss each of the underlying technologies to create and implement an e-commerce website.

TABLE OF CONTENTS

1. INTRODUCTION.....	1
2. LITERATURE REVIEW	1
3. PROJECT DESIGN.....	3
3.1 DATA MODEL.....	4
3.1.1 Database Design.....	6
3.2. PROCESS MODEL	10
3.2.1. Functional Decomposition Diagram.....	10
3.2.2 Data Flow Diagram (DFD).....	12
3.3 USER INTERFACE DESIGN	19
4. IMPLEMENTATION TECHNOLOGIES	23
4.1. INTERNET INFORMATION SERVICES (IIS)	24
4.2 ASP.NET	25
4.2.1. Authentication in ASP.NET	26
4.3. MYSQL DATABASE	27
4.4. INTEGRATING IIS AND ASP.NET	28
4.5. INTEGRATING THE WEBSITE AND DATABASE.....	29
5. WEB PAGE PROGRAMMING OPTIONS.....	30
5.1. SERVER-SIDE PROCESSING.	31
5.2. CLIENT-SIDE PROCESSING.	35
6. WEB BASED APPLICATION DEVELOPMENT.....	36
7. DATABASE CONNECTIVITY	38
7.1 ADO.NET.....	38
7.2 CONNECTING ASP.NET APPLICATION TO A DATABASE.....	45
8. THE SHOPPING CART APPLICATION	46
8.1. SEARCH FOR BOOKS	48
8.2. REGISTRATION.....	49
8.3. USER DETAILS	50
8.4. SHOPPING CART	51
8.5. PLACE AN ORDER	52
8.6. CHECK OUT	54
8.7. PURCHASE HISTORY.....	56
8.8. TRANSACTIONS IN THE APPLICATION.....	61
9. LIMITATIONS AND FUTURE DEVELOPMENT.....	65
10. CONCLUSION	65
11. BIBLIOGRAPHY	67

LIST OF FIGURES

FIGURE 1 ENTITY RELATIONSHIP DIAGRAM (ERD)	4
FIGURE 2 FUNCTIONAL DECOMPOSITION DIAGRAM.....	11
FIGURE 3 CUSTOMER - BROWSE CONTEXT DFD.....	13
FIGURE 4 CUSTOMER - BROWSE DETAILED DFD.....	13
FIGURE 5 CUSTOMER – SHOPPING CART CONTEXT DFD	14
FIGURE 6 CUSTOMER - SHOPPING CART DETAILED DFD.....	14
FIGURE 7 CUSTOMER - SHOPPING CART DETAILED DFD.....	15
FIGURE 8 CUSTOMER - SHOPPING CART DETAILED DFD.....	15
FIGURE 9 CUSTOMER – AUTHENTICATION – USERPROFILE DFD.....	16
FIGURE 10 AUTHENTICATED USER-PURCHASE CONTEXT DFD	16
FIGURE 11 CUSTOMER - AUTHENTICATION - PURCHASE DFD	17
FIGURE 12 CUSTOMER - NEWUSERREGISTRATION DFD	18
FIGURE 13 ADMINSTRATOR CONTEXT DFD	18
FIGURE 14 ADMINISTRATOR DETAILED DFD	18
FIGURE 15 MENU	19
FIGURE 16 DISPLAY OF BOOKS PRESENT IN THE STORE.....	19
FIGURE 17 FOR SEARCHING THE BOOKS IN THE STORE	20
FIGURE 18 SHOPPING CART FOR THE USER.....	20
FIGURE 19 REGISTRATION OF THE NEW USER.....	21
FIGURE 20 AUTHENTICATION OF THE USER	22
FIGURE 21 RELATION BETWEEN IIS AND ASP.NET.....	23
FIGURE 22 WEB PAGE PROGRAMMING OPTIONS	30
FIGURE 23 COMPILED SERVER PROGRAMS FLOWCHART.....	31
FIGURE 24 ADO.NET ARCHITECTURE	43
FIGURE 25 BOOK DETAILS	47
FIGURE 26 SERACH FOR BOOKS.....	48
FIGURE 27 NEW USER REGISTRATION.....	49
FIGURE 28 USER DETAILS	50
FIGURE 29 SHOPPING CART	51
FIGURE 30 ORDER DETAILS	52
FIGURE 31 SHIPPING DETAILS	53
FIGURE 32 CHECK OUT	54
FIGURE 33 ORDER CONFIRMATION.....	55
FIGURE 34 UPDATED INVENTORY AFTER ORDER PLACEMENT	55
FIGURE 35 PURCHASE HISTORY.....	56
FIGURE 36 BOOK DETAILS	57
FIGURE 37 ADMINISTRATOR - MODIFY BOOKS.....	58
FIGURE 38 DETAILS ABOUT NEW BOOK	59
FIGURE 39 UPDATED INVENTORY	60

LIST OF TABLES

TABLE 1 PROCESSING TECHNOLOGY FOR DIFFERENT FILE EXTENSIONS.....	33
TABLE 2 TRANSACTION ATTRIBUTES	64

1. Introduction

E-commerce is fast gaining ground as an accepted and used business paradigm. More and more business houses are implementing web sites providing functionality for performing commercial transactions over the web. It is reasonable to say that the process of shopping on the web is becoming commonplace.

The objective of this project is to develop a general purpose e-commerce store where any product (such as books, CDs, computers, mobile phones, electronic items, and home appliances) can be bought from the comfort of home through the Internet. However, for implementation purposes, this paper will deal with an online book store.

An online store is a virtual store on the Internet where customers can browse the catalog and select products of interest. The selected items may be collected in a shopping cart. At checkout time, the items in the shopping cart will be presented as an order. At that time, more information will be needed to complete the transaction. Usually, the customer will be asked to fill or select a billing address, a shipping address, a shipping option, and payment information such as credit card number. An e-mail notification is sent to the customer as soon as the order is placed.

2. Literature Review

Electronic Commerce (e-commerce) applications support the interaction between different parties participating in a commerce transaction via the network, as well as the management of the data involved in the process [2].

The increasing importance of e-commerce is apparent in the study conducted by researches at the GVI (Graphics, Visualization, and Usability) Center at the Georgia Institute of Technology. In their summary of the findings from the eighth survey, the researchers report that “e-commerce is taking off both in terms of the number of users shopping as well as the total amount people are spending via Internet based transactions”.

Over three quarters of the 10,000 respondents report having purchased items online. The most cited reason for using the web for personal shopping was convenience (65%), followed by availability of vendor information (60%), no pressure from sales person (55%) and saving time (53%).

Although the issue of security remains the primary reason why more people do not purchase items online, the GVA survey also indicates that faith in the security of e-commerce is increasing. As more people gain confidence in current encryption technologies, more and more users can be expected to frequently purchase items online [11].

A good e-commerce site should present the following factors to the customers for better usability [11]:

- Knowing when an item was saved or not saved in the shopping cart.
- Returning to different parts of the site after adding an item to the shopping cart.
- Easy scanning and selecting items in a list.
- Effective categorical organization of products.
- Simple navigation from home page to information and order links for specific products.
- Obvious shopping links or buttons.
- Minimal and effective security notifications or messages.
- Consistent layout of product information.

Another important factor in the design of an e-commerce site is feedback [4]. The interactive cycle between a user and a web site is not complete until the web site responds to a command entered by the user. According to Norman [5], "feedback--sending back to the user information about what action has actually been done, what result has been accomplished--is a well known concept in the science of control and information theory. Imagine trying to talk to someone when you cannot even hear your own voice, or trying to draw a picture with a pencil that leaves no mark: there would be no feedback".

Web site feedback often consists of a change in the visual or verbal information presented to the user. Simple examples include highlighting a selection made by the user or filling a field on a form based on a user's selection from a pull down list. Another example is using the sound of a cash register to confirm that a product has been added to an electronic shopping cart.

Completed orders should be acknowledged quickly. This may be done with an acknowledgment or fulfillment page. The amount of time it takes to generate and download this page, however, is a source of irritation for many e-commerce users. Users are quick to attribute meaning to events. A blank page, or what a user perceives to be "a long time" to receive an acknowledgment, may be interpreted as "there must be something wrong with the order." If generating an acknowledgment may take longer than what may be reasonably expected by the user, then the design should include intermediate feedback to the user indicating the progress being made toward acknowledgment or fulfillment.

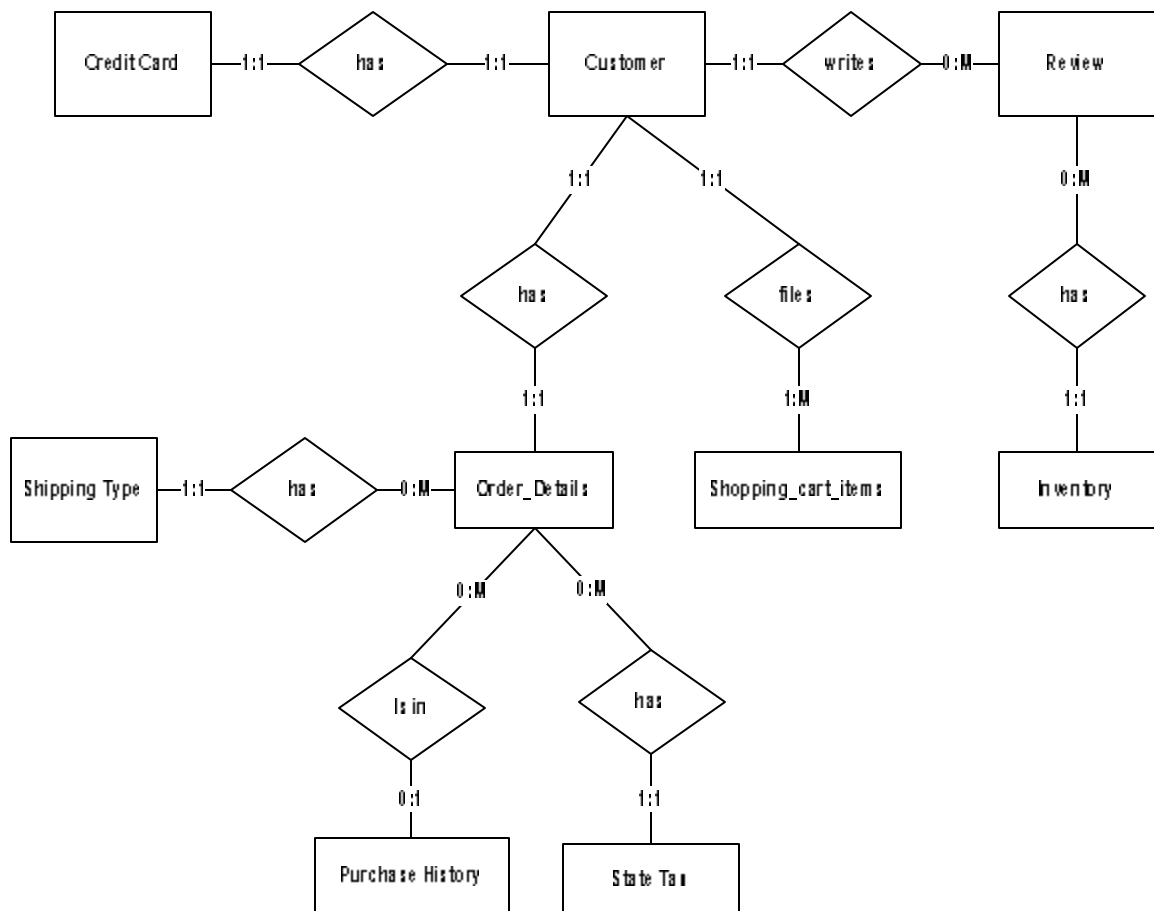
Finally, feedback should not distract the user. Actions and reactions made by the web site should be meaningful. Feedback should not draw the user's attention away from the important tasks of gathering information, selecting products, and placing orders.

3. Project Design

In order to design a web site, the relational database must be designed first. Conceptual design can be divided into two parts: The **data model** and the **process model**. The data model focuses on what data should be stored in the database while the process model deals with how the data is processed. To put this in the context of the relational database, the data model is used to design the relational tables. The process model is used to design the queries that will access and perform operations on those tables.

3.1 Data Model

A data model is a conceptual representation of the data structures that are required by a database. The first step in designing a database is to develop an Entity-Relation Diagram (ERD). The ERD serves as a blue print from which a relational database maybe deduced. Figure 1 shows the ERD for the project and later we will show the transformation from ERD to the Relational model.



entity A matches exactly one record in entity B and every record in B matches exactly one record in A. One to many means that every record in A matches zero or more records in B and every record in B matches exactly one record in A. If there is a one to many relationship between two entities, then these entities are represented as Associative Entities. In the Relational Database model, each of the entities will be transformed into a table. The tables are shown below along with the attributes.

3.1.1 Database Design

In this section, the basic structure of the tables composing the database for the project are shown along with information about primary and foreign keys.

Customer

SNO	NAME	TYPE	DESCRIPTION
1	<u>UserID</u>	Varchar	Primary key for Customer identification
2	Password	Varchar	Security for Customer
3	First_Name	Varchar	
4	Last_Name	Varchar	
5	Address	Varchar	
6	City	Varchar	
7	Zip	Integer	
8	State	Varchar	
9	Email Address	Varchar	
10	Phone_Number	Varchar	

Books

SNO	NAME	TYPE	DESCRIPTION
1	<u>InventoryID</u>	Integer	Primary key for Inventory Identification, ISBN of a book
2	Book_Name	Varchar	
3	Author	Varchar	
5	Nr_books	Integer	
6	Price	Double	

State_Tax

SNO	NAME	TYPE	DESCRIPTION
1	<u>State Name</u>	Varchar	Primary key for State Identification
2	Sales Tax Rate	Double	Sales tax for each state

Shopping_Cart_Items

SNO	NAME	TYPE	DESCRIPTION
1	<u>ShoppingCartID</u>	Integer	Primary key for Shopping Cart Identification
2	InventoryID	Varchar	Foreign key to Inventory
3	Price	Double	
4	Date	Date	
5	UserID	Varchar	Foreign key to Customer
6	Quantity	Integer	

Order_Details

SNO	NAME	TYPE	DESCRIPTION
1	OrderID	Integer	Primary key for Order identification
1	UserID	Char	Foreign key to Customer
2	Receiver's Name	Char	If order is to be sent to other address rather than to the customer, we need that address
3	Address	Char	
4	City	Char	
5	Zip	Integer	
6	State	Char	Foreign key to State Tax
7	Type of Shipping	Char	Foreign key to Shipping Type
8	Date of Purchase	Date	

Shipping_Type

SNO	NAME	TYPE	DESCRIPTION
1	<u>Type of Shipping</u>	Varchar	Primary key to define type of shipping
2	Price	Double	
3	Approximate days for delivery	Integer	

Credit_Card_Details

SNO	NAME	TYPE	DESCRIPTION
1	<u>Credit Username</u>	Varchar	Primary key for Customer Identification
2	Credit Card Number	Varchar	
3	Card Type	Varchar	Master Card, Visa, Discover
4	CVV Number	Integer	Number present on the back of the card for extra security
5	Expiry Date	Date	
6	UserID	Varchar	Foreign key to Customer

Book_Review

SNO	NAME	TYPE	DESCRIPTION
1	InventoryID	Varchar	ISBN of the book on which the review is written
2	Reviews	Varchar	Review on the book
3	Rating	Varchar	Rating given to the book in a scale of 5
4	Review Date	Date	
5	User Name	Varchar	Name of the user providing the review

Purchase_History

SNO	NAME	TYPE	DESCRIPTION
1	<u>UserID</u>	Varchar	Primary key for Customer Identification
2	InventoryID	Varchar	Book purchased by the user
3	Date of Purchase	Date	
4	OrderID	Integer	Foreign key to Order_details
5	Quantity	Integer	
6	Price	Double	

3.2. Process Model

A Process Model tells us about how the data is processed and how the data flows from one table to another to gather the required information. This model consists of the Functional Decomposition Diagram and Data Flow Diagram.

3.2.1. Functional Decomposition Diagram

A decomposition diagram shows a top-down functional decomposition of a system and exposes the system's structure. The objective of the Functional Decomposition is to break down a system step by step, beginning with the main function of a system and continuing with the interim levels down to the level of elementary functions. The diagram is the starting point for more detailed process diagrams, such as data flow diagrams (DFD). Figure 2 shows the Functional Decomposition Diagram for this project.

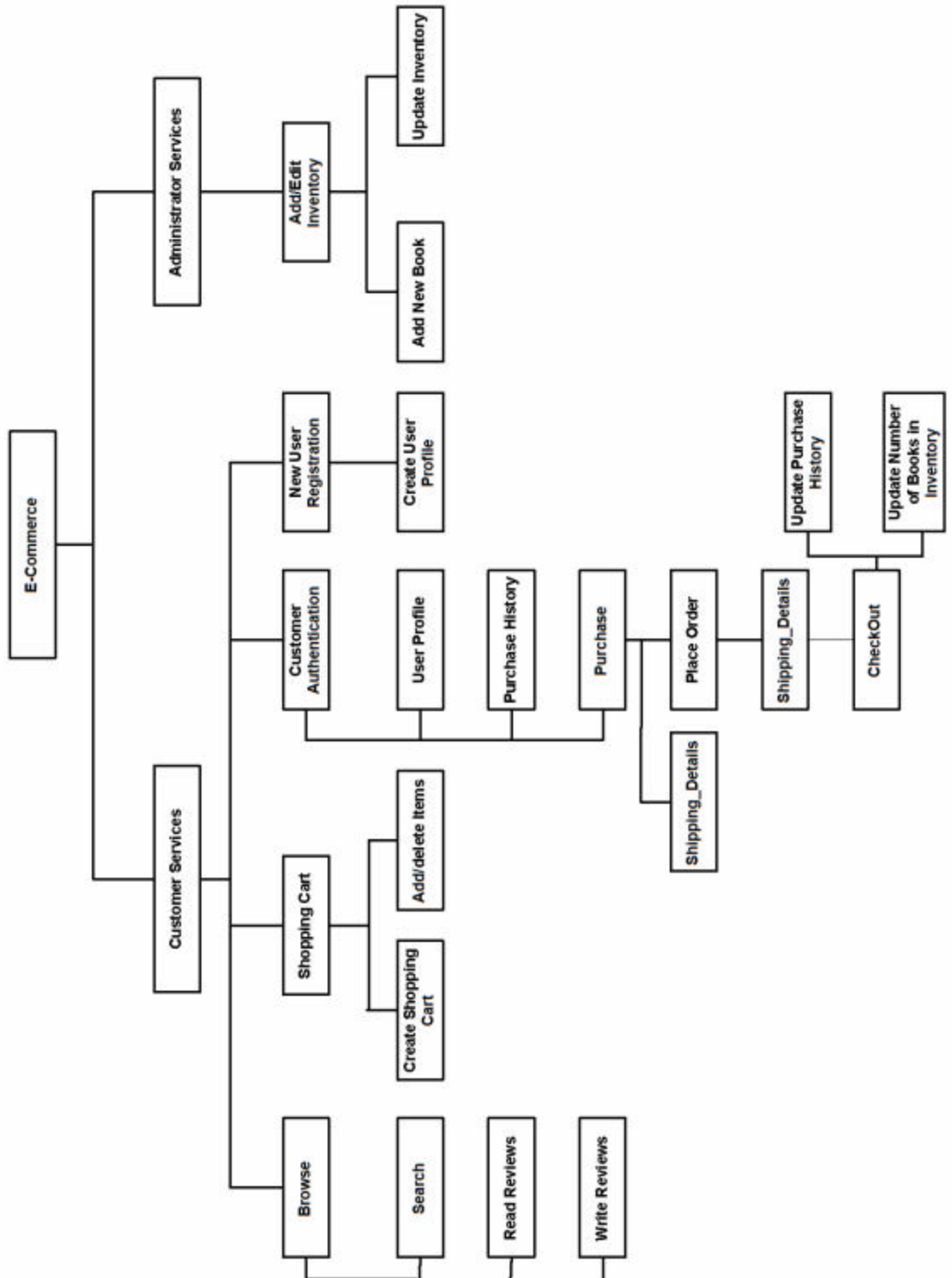


Figure 2 Functional Decomposition Diagram

3.2.2 Data Flow Diagram (DFD)

Data Flow Diagrams show the flow of data from external entities into the system, and from one process to another within the system. There are four symbols for drawing a DFD:

1. Rectangles representing *external entities*, which are sources or destinations of data.
2. Ellipses representing *processes*, which take data as input, validate and process it and output it.
3. Arrows representing the *data flows*, which can either, be electronic data or physical items.
4. Open-ended rectangles or a Disk symbol representing *data stores*, including electronic stores such as databases or XML files and physical stores such as filing cabinets or stacks of paper.

Figures 3 - 14 are the Data Flow Diagrams for the current system. Each process within the system is first shown as a Context Level DFD and later as a Detailed DFD. The Context Level DFD provides a conceptual view of the process and its surrounding input, output and data stores. The Detailed DFD provides a more detailed and comprehensive view of the interaction among the sub-processes within the system.

Customer-Browse Context DFD

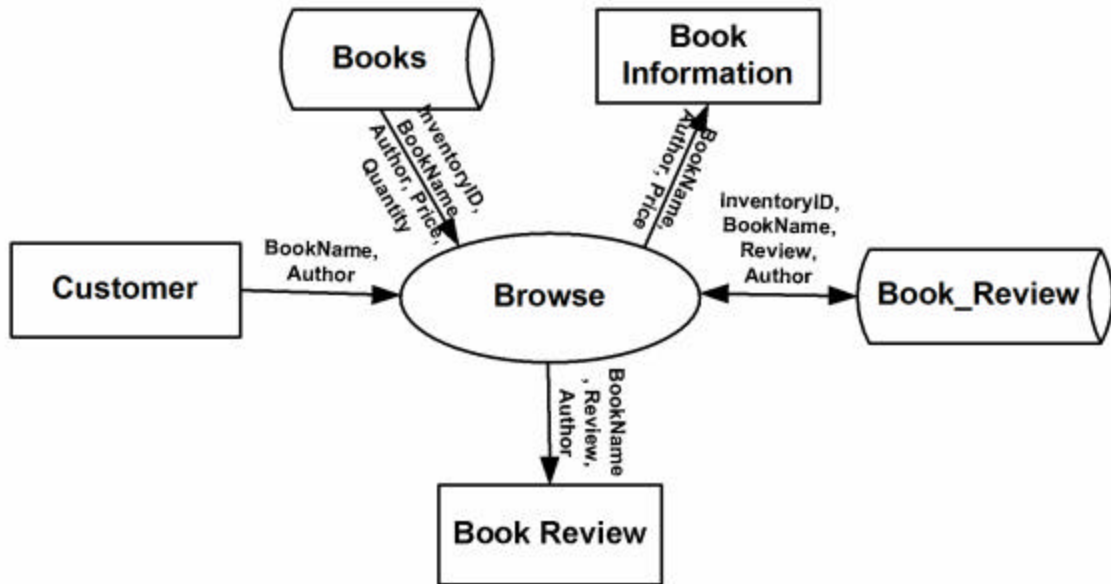


Figure 3 Customer - Browse Context DFD

Customer-Browse Detailed DFD

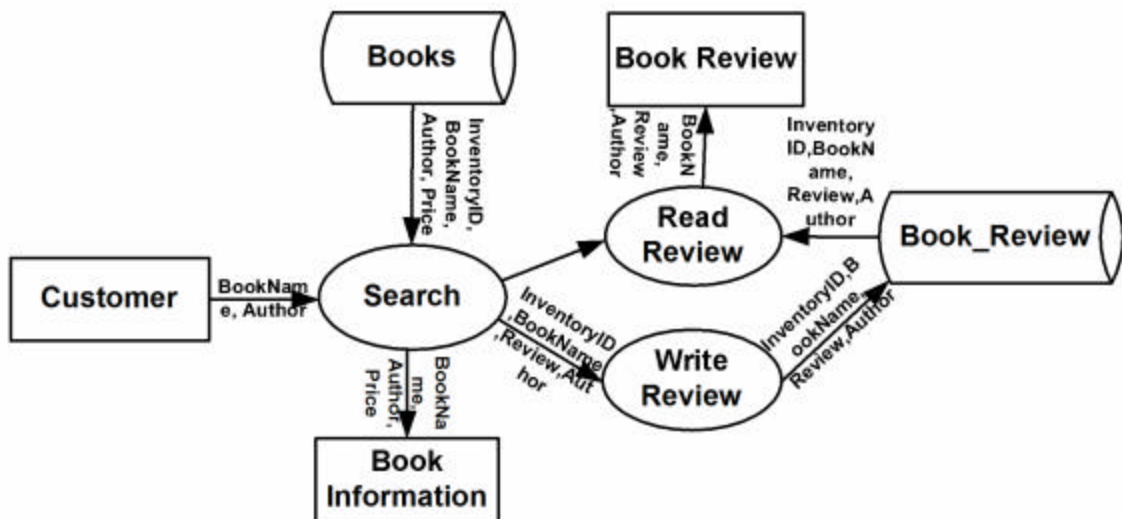


Figure 4 Customer - Browse Detailed DFD

Customer - ShoppingCart Context DFD

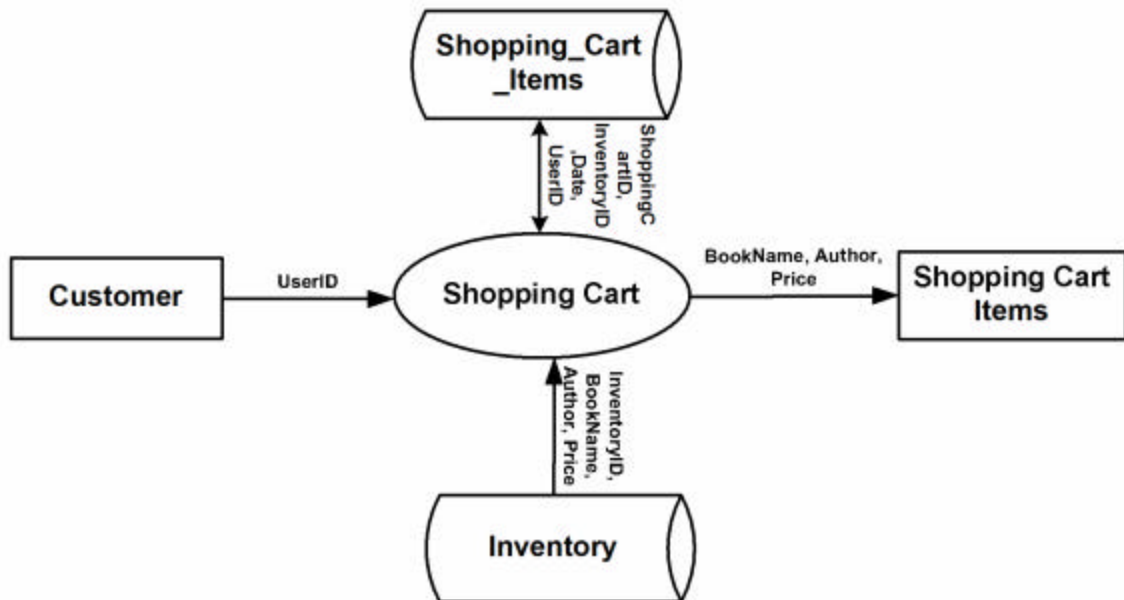


Figure 5 Customer – Shopping Cart Context DFD

Customer - ShoppingCart Detailed DFD

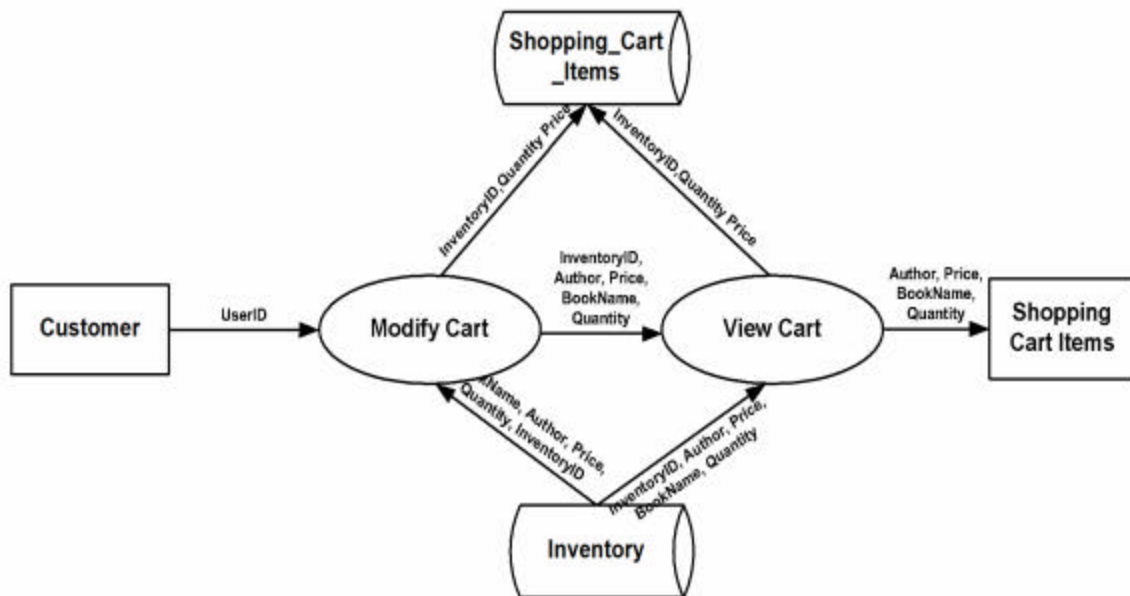


Figure 6 Customer - Shopping Cart Detailed DFD

Customer-Authentication Context DFD

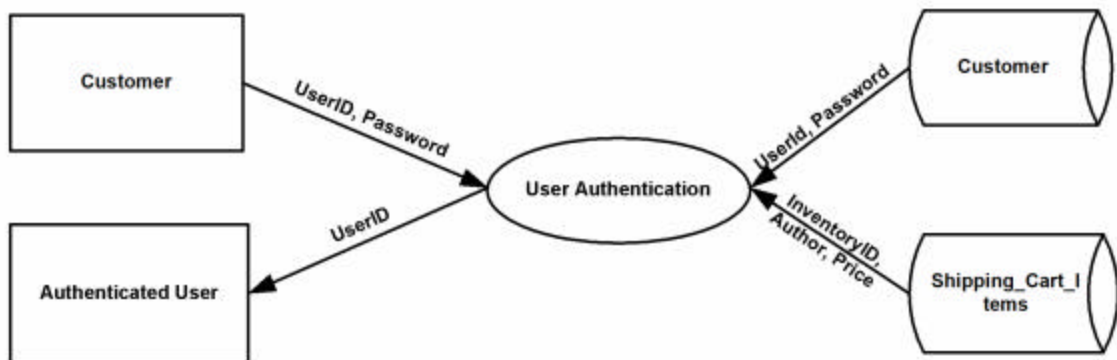


Figure 7 Customer - Shopping Cart Detailed DFD

Customer-Authentication-PurchaseHistory DFD

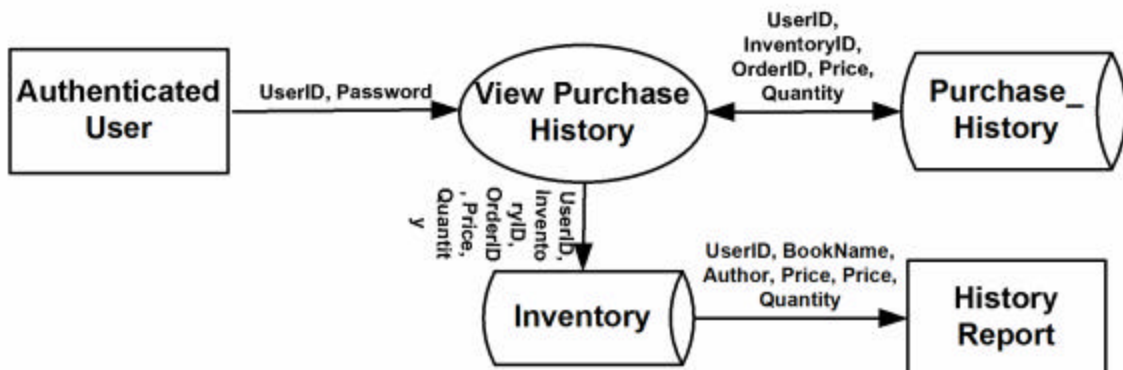


Figure 8 Customer - Shopping Cart Detailed DFD

Customer-Authentication-UserProfile DFD



Figure 9 Customer – Authentication – UserProfile DFD

Authenticated User-Purchase Context DFD

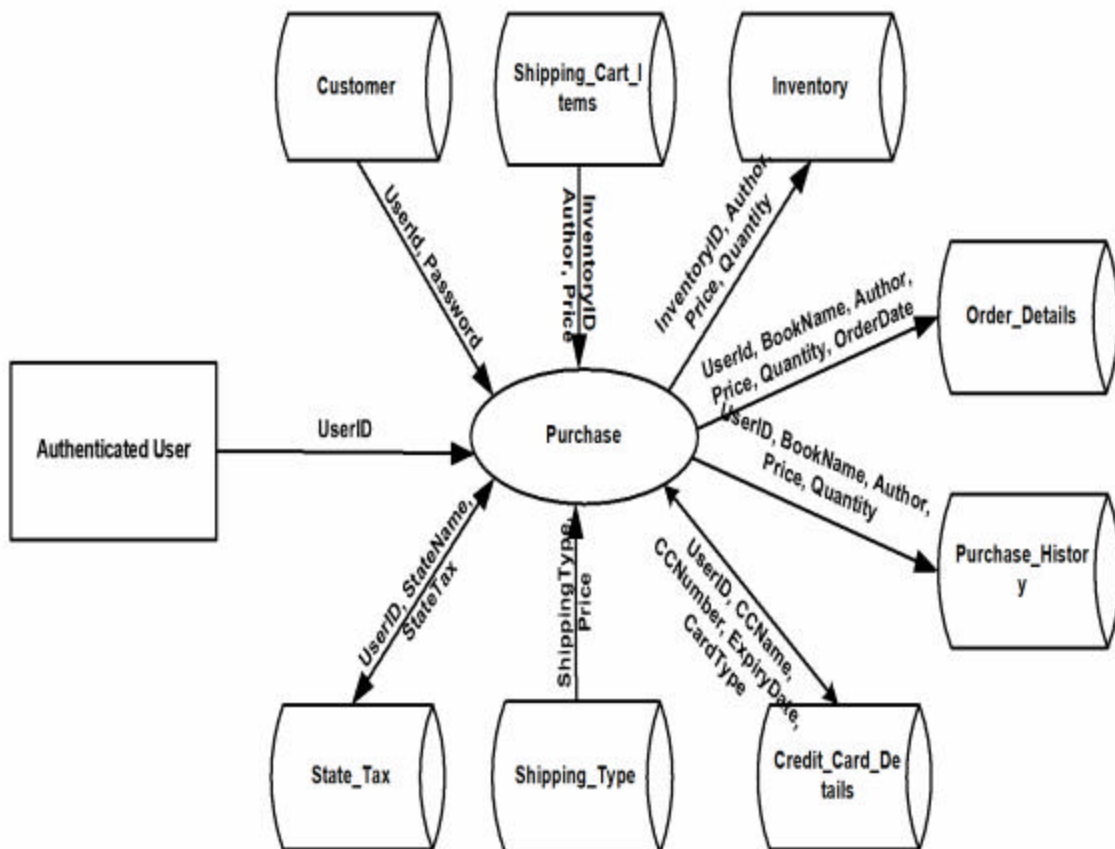


Figure 10 Authenticated User-Purchase Context DFD

Authenticated User-Purchase DFD

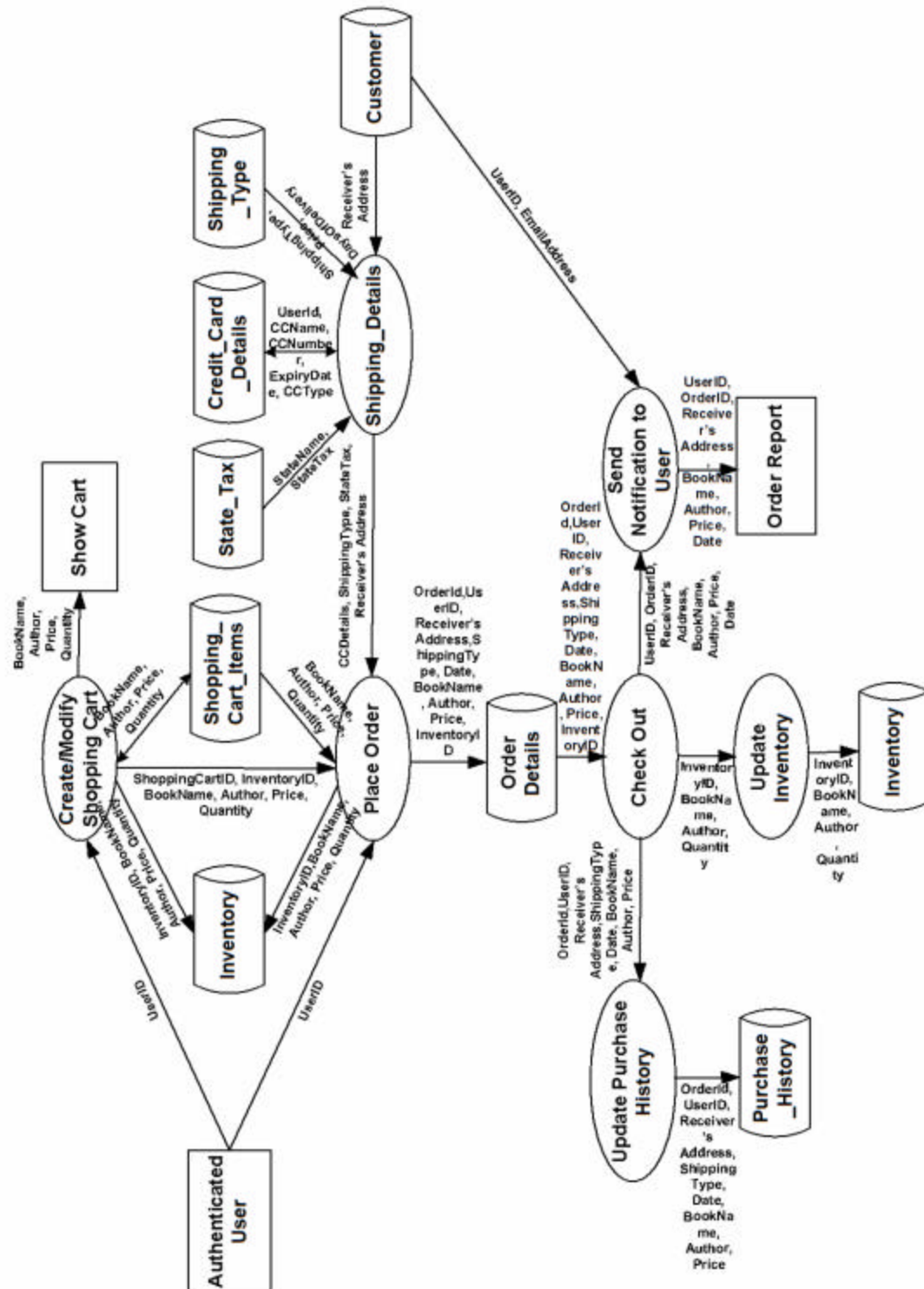
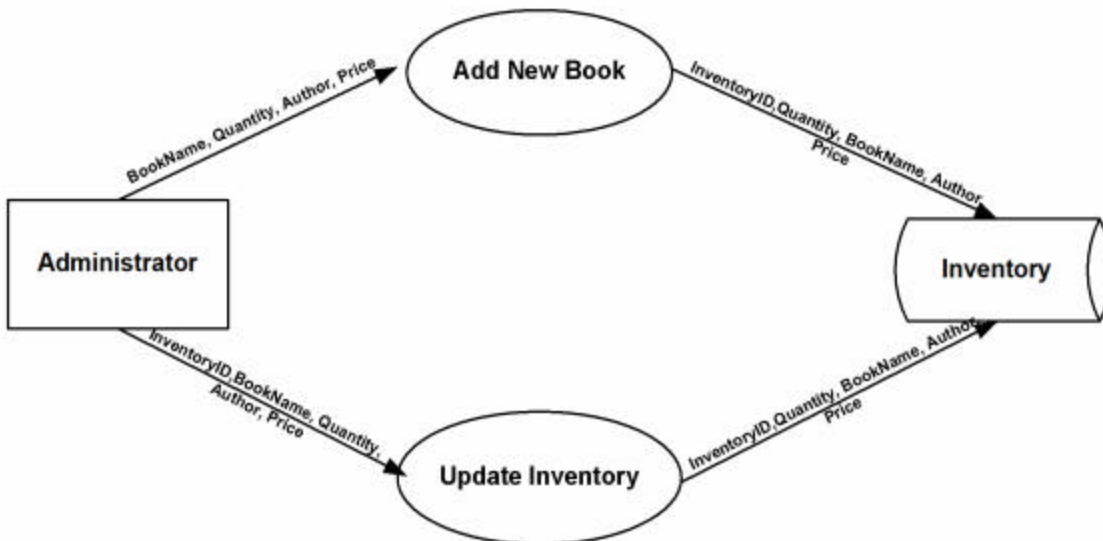
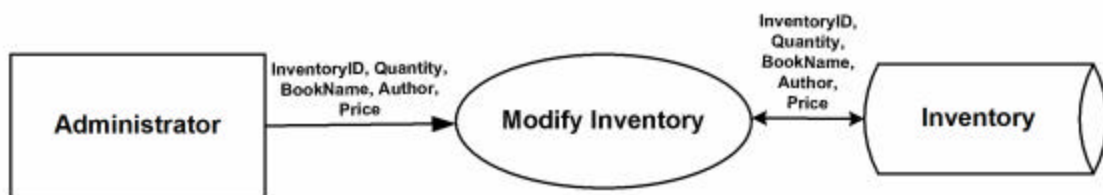


Figure 11 Customer - Authentication - Purchase DFD

Customer-NewUserRegistration DFD



3.3 User Interface Design

Before implementing the actual design of the project, a few user interface designs were constructed to visualize the user interaction with the system as they browse for books, create a shopping cart and purchase books. The user interface design will closely follow our Functional Decomposition Diagram (Figure 2). Figures 15 – 20 show the initial designs of the web pages.

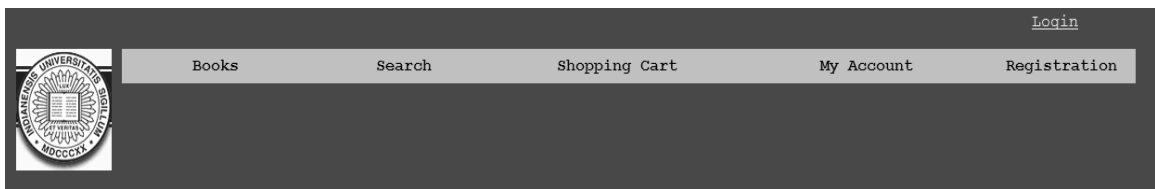
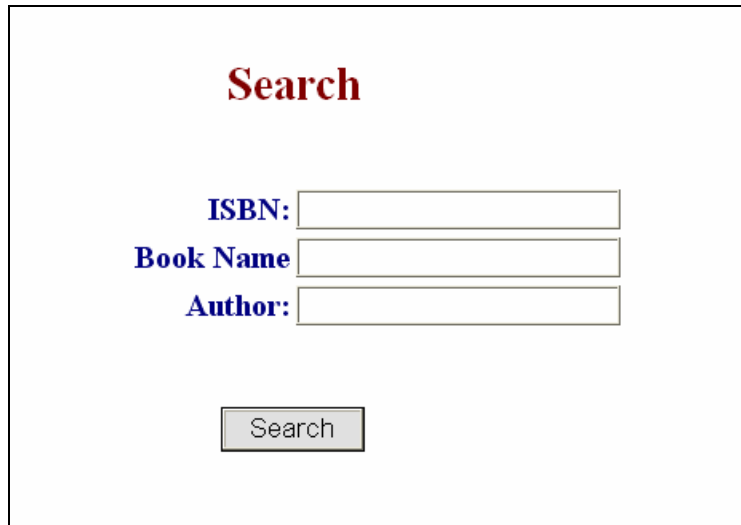


Figure 15 Menu

Book Details				
Book Title	Author	Quantity Available	Price	Add to Cart

Figure 16 Display of Books present in the store



A search form titled "Search" in a large, bold, dark red font. Below the title are three input fields, each preceded by a label in a bold, dark blue font: "ISBN:", "Book Name", and "Author:". The input fields are simple rectangular boxes. Below these fields is a single button with the text "Search" in a standard black font.

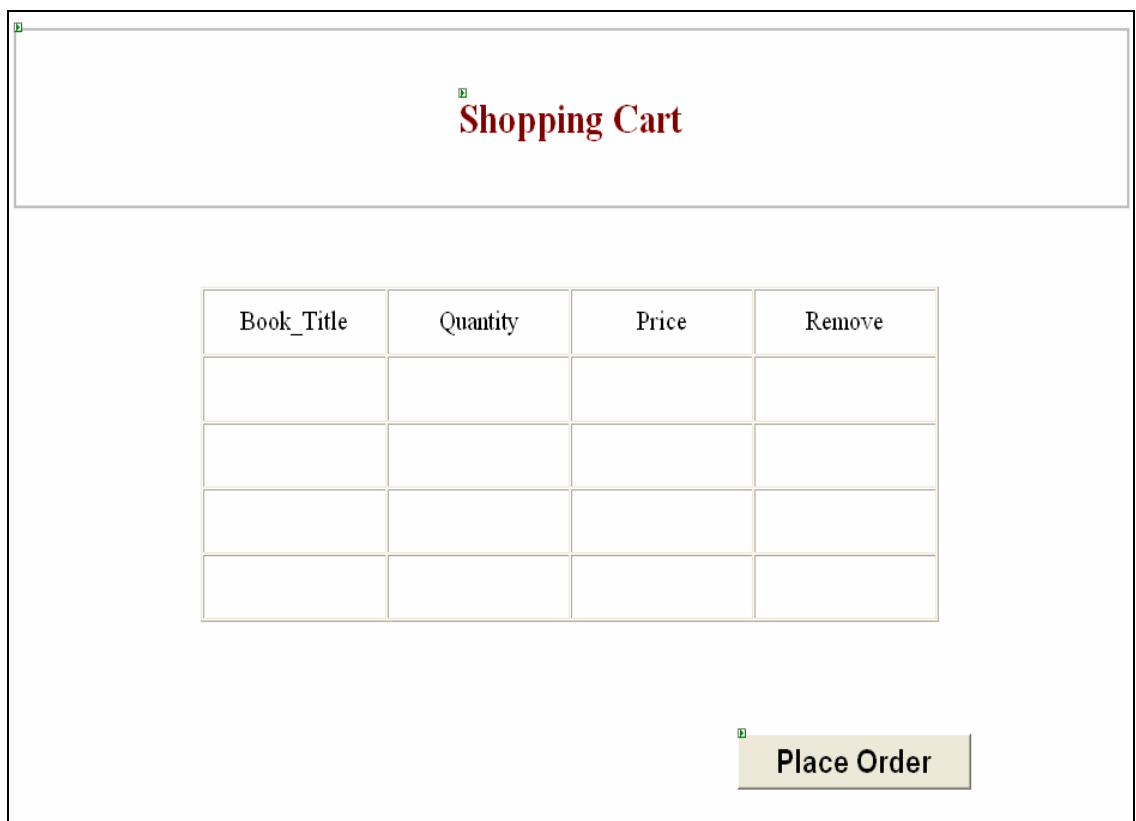
Search

ISBN:

Book Name

Author:

Figure 17 For searching the books in the store



A shopping cart interface. At the top, the title "Shopping Cart" is displayed in a large, bold, dark red font. Below the title is a table with four columns: "Book_Title", "Quantity", "Price", and "Remove". The table has four empty rows for data entry. At the bottom right of the interface is a button labeled "Place Order".

Shopping Cart

Book_Title	Quantity	Price	Remove

Figure 18 Shopping Cart for the user

Registration

[Login](#)

User ID*	<input type="text"/>
Password*	<input type="password"/>
Confirm Password*	<input type="password"/>
First Name*	<input type="text"/>
Last Name	<input type="text"/>
Address*	<input type="text"/>
City*	<input type="text"/>
Zip*	<input type="text"/>
State*	<input type="text" value="SELECT"/>
Email*	<input type="text"/>
Daytime Phone Number	<input type="text"/>
<div><input type="button" value="Submit"/> <input type="button" value="Reset"/></div>	

Figure 19 Registration of the new user

The image shows a login interface within a black rectangular border. At the top, the word "Login" is centered in a large, bold, dark red serif font. Below it, the message "Incorrect Password" is centered in a smaller, bold, red sans-serif font. Further down, there are two input fields: the first is preceded by the label "User ID:" and the second by "Password:", both in a bold, dark blue sans-serif font. The input fields themselves are simple white rectangles with thin black borders. Below the input fields, there are two blue, underlined links: "New User" on the left and "Forgot Password?" on the right. At the bottom center, there is a light gray rectangular button with the word "Enter" in a dark gray sans-serif font.

Figure 20 Authentication of the user

4. Implementation Technologies

The objective of this project is to develop an online book store. When the user types in the URL of the Book Store in the address field of the browser, a Web Server is contacted to get the requested information. In the .NET Framework, IIS (Internet Information Service) acts as the Web Server. The sole task of a Web Server is to accept incoming HTTP requests and to return the requested resource in an HTTP response. The first thing IIS does when a request comes in is to decide how to handle the request. Its decision is based upon the requested file's extension. For example, if the requested file has the .asp extension, IIS will route the request to be handled by asp.dll. If it has the extension of .aspx, .ascx, etc, it will route the request to be handled by ASP.NET Engine.

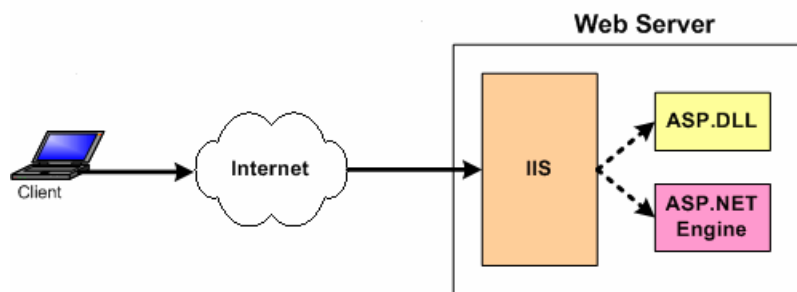


Figure 21 Relation between IIS and ASP.NET

The ASP.NET Engine then gets the requested file, and if necessary contacts the database through ADO.NET for the required file and then the information is sent back to the Client's browser. Figure 21 shows how a client browser interacts with the Web server and how the Web server handles the request from client.

4.1. Internet Information Services (IIS)

IIS is a set of Internet based services for Windows machines. Originally supplied as part of the Option Pack for Windows NT, they were subsequently integrated with Windows 2000 and Windows Server 2003). The current (Windows 2003) version is IIS 6.0 and includes servers for **FTP** (a software standard for transferring computer files between machines with widely different operating systems), **SMTP** (Simple Mail Transfer Protocol, is the de facto standard for email transmission across the Internet) and **HTTP/HTTPS** (is the secure version of HTTP, the communication protocol of the World Wide Web) [12].

Features: The web server itself cannot directly perform server side processing but can delegate the task to ISAPI (Application Programming Interface of IIS) applications on the server. Microsoft provides a number of these including ones for Active Server Page and ASP.NET.

Compatibility: Internet Information Services is designed to run on Windows server operating systems. A restricted version that supports one web site and a limited number of connections is also supplied with Windows XP Professional.

Microsoft has also changed the server account that IIS runs on. In versions of IIS before 6.0, all the features were run on the System account, allowing exploits to run wild on the system. Under 6.0 many of the processes have been brought under a Network Services account that has fewer privileges. In particular this means that if there were an exploit on that feature, it would not necessarily compromise the entire system.

4.2 ASP.NET

ASP.NET is a programming framework built on the common language runtime that can be used on a server to build powerful Web applications. ASP.NET has many advantages – both for programmers and for the end users because it is compatible with the .NET Framework. This compatibility allows the users to use the following features through ASP.NET:

- a) **Powerful database-driven functionality:** ASP.NET allows programmers to develop web applications that interface with a database. The advantage of ASP.NET is that it is object-oriented and has many programming tools that allow for faster development and more functionality.
- b) **Faster web applications:** Two aspects of ASP.NET make it fast -- compiled code and caching. In ASP.NET the code is compiled into "machine language" *before* a visitor ever comes to the website. Caching is the storage of information in memory for faster access in the future. ASP.NET allows programmers to set up pages or areas of pages that are commonly reused to be cached for a set period of time to improve the performance of web applications. In addition, ASP.NET allows the caching of data from a database so the website is not slowed down by frequent visits to a database when the data does not change very often.
- c) **Memory leak and crash protection:** ASP.NET automatically recovers from memory leaks and errors to make sure that the website is always available to the visitors.

ASP.NET also supports code written in more than 25 .NET languages (including VB.NET, C#, and Jscript.Net). This is achieved by the Common Language Runtime (CLR) compiler that supports multiple languages.

4.2.1. Authentication in ASP.NET

There are two separate authentication layers in an ASP.NET application. All requests flow through IIS before they are handed to ASP.NET, and IIS can decide to deny access before ASP.NET even knows about the request. Here is how the process works [14]:

1. IIS checks to see if an incoming request is coming from an IP address that is allowed access to the domain. If not, the request is denied.
2. IIS performs its own user authentication, if it is configured to do so. By default, IIS allows anonymous access and requests are authenticated automatically.
3. When a request is passed from IIS to ASP.NET with an authenticated user, ASP.NET checks to see whether impersonation is enabled. If so, ASP.NET acts as though it were the authenticated user. If not, ASP.NET acts with its own configured account.
4. Finally, the identity is used to request resources from the operating system. If all the necessary resources can be obtained, the user's request is granted; otherwise the request is denied.

4.3. MySQL Database

In this project, MySQL is used as the backend database. MySQL is an open-source database management system. The features of MySQL are given below:

- MySQL is a relational database management system. A relational database stores information in different tables, rather than in one giant table. These tables can be referenced to each other, to access and maintain data easily.
- MySQL is open source database system. The database software can be used and modify by anyone according to their needs.
- It is fast, reliable and easy to use. To improve the performance, MySQL is multi-threaded database engine. A multithreaded application performs many tasks at the same time as if multiple instances of that application were running simultaneously.

In being multithreaded MySQL has many advantages. A separate thread handles each incoming connection with an extra thread that is always running to manage the connections. Multiple clients can perform read operations simultaneously, but while writing, only hold up another client that needs access to the data being updated. Even though the threads share the same process space, they execute individually and because of this separation, multiprocessor machines can spread the thread across many CPUs as long as the host operating system supports multiple CPUs. Multithreading is the key feature to support MySQL's performance design goals. It is the core feature around which MySQL is built.

MySQL database is connected to ASP.NET using an ODBC driver. Open Database Connectivity (ODBC) is a widely accepted application-programming interface (API) for database access. The ODBC driver is a library that implements the functions supported by ODBC API. It processes ODBC function calls, submits SQL requests to MySQL server, and returns results back to the application. If necessary, the driver modifies an application's request so that the request conforms to syntax supported by MySQL.

4.4. Integrating IIS and ASP.NET

When a request comes into IIS Web server its extension is examined and, based on this extension, the request is either handled directly by IIS or routed to an ISAPI extension. An ISAPI extension is a compiled class that is installed on the Web server and whose responsibility is to return the markup for the requested file type. By default, IIS handles the request, and simply returns the contents of the requested file [13].

This makes sense for static files, like images, HTML pages, CSS files, external JavaScript files, and so on. For example, when a request is made for a .html file, IIS simply returns the contents of the requested HTML file.

For files whose content is dynamically generated, the ISAPI extension configured for the file extension is responsible for generating the content for the requested file. For example, a Web site that serves up classic ASP pages has the .asp extension mapped to the asp.dll ISAPI extension. The asp.dll ISAPI extension executes the requested ASP page and returns its generated HTML markup. If the Web site serves up ASP.NET Web pages, IIS has mapped the .aspx to aspnet_isapi.dll, an ISAPI extension that starts off the process of generating the rendered HTML for the requested ASP.NET Web page.

The aspnet_isapi.dll ISAPI extension is a piece of *unmanaged code*. That is, it is not code that runs in the .NET Framework. When IIS routes the request to the aspnet_isapi.dll ISAPI extension, the ISAPI extension routes the request onto the ASP.NET engine, which is written in *managed code* - managed code is code that runs in the .NET Framework.

The ASP.NET engine is strikingly similar to IIS in many ways. Just like IIS has a directory mapping file extensions to ISAPI extensions, the ASP.NET engine maps file extensions to *HTTP handlers*. An HTTP handler is a piece of managed code that is responsible for generating the markup for a particular file type.

4.5. Integrating the Website and Database

Customers ordering from an e-commerce website need to be able to get information about a vendor's products and services, ask questions, select items they wish to purchase, and submit payment information. Vendors need to be able to track customer inquiries and preferences and process their orders. So a well organized database is essential for the development and maintenance of an e-commerce site [3].

In a static Web page, content is determined at the time when the page is created. As users access a static page, the page always displays the same information. Example of a static Web page is the page displaying company information. In a dynamic Web page, content varies based on user input and data received from external sources. We use the term "data-based Web pages" to refer to dynamic Web pages deriving some or all of their content from data files or databases.

A data-based Web page is requested when a user clicks a hyperlink or the submit button on a Web page form. If the request comes from clicking a hyperlink, the link specifies either a Web server program or a Web page that calls a Web server program. In some cases, the program performs a static query, such as "Display all items from the Inventory". Although this query requires no user input, the results vary depending on when the query is made. If the request is generated when the user clicks a form's submit button, instead of a hyperlink, the Web server program typically uses the form inputs to create a query. For example, the user might select five books to be purchased and then submit the input to the Web server program. The Web server program then services the order, generating a dynamic Web page response to confirm the transaction. In either case, the Web server is responsible for formatting the query results by adding HTML tags. The Web server program then sends the program's output back to the client's browser as a Web page.

5. Web Page Programming Options

An e-commerce organization can create data-based Web pages by using server-side and client-side processing technologies or a hybrid of the two. With server-side processing, the Web server receives the dynamic Web page request, performs all processing necessary to create the page, and then sends it to the client for display in the client's browser. Client-side processing is done on the client workstation by having the client browser execute a program that interacts directly with the database.

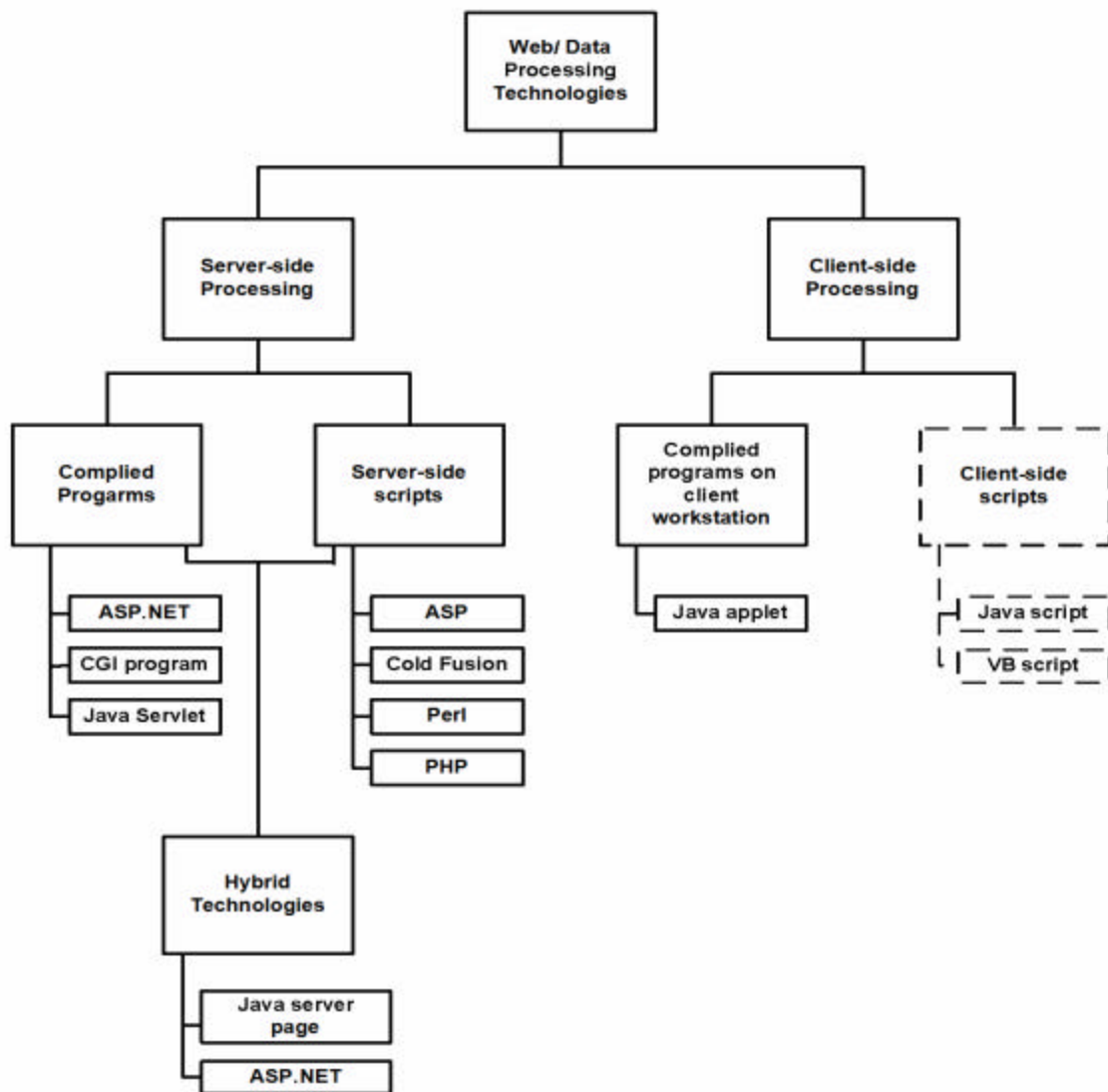


Figure 22 Web page programming options

Figure 22 (reproduced from [3]) outlines commonly used server-side, client-side, and hybrid Web and data processing technologies; client-side scripts are in dashed lines to indicate they are unable to interact directly with a database or file but are used to validate user input on the client, then send the validated inputs to the server for further processing.

5.1. Server-side processing.

Generally dynamic or data-driven Web pages use HTML forms to collect user inputs, submitting them to a Web server. A program running on the server processes the form inputs, dynamically composing a Web page reply. This program, which is called, servicing program, can be either a compiled executable program or a script interpreted into machine language each time it is run.

Compiled server programs. When a user submits HTML-form data for processing by a compiled server program, the Web Server invokes the servicing program. The servicing program is not part of the Web server but it is an independent executable program running on the Web server; it processes the user input, determines the action which must be taken, interacts with any external sources (Eg: database) and finally produces an HTML document and terminates. The Web server then sends the HTML document back to the user's browser where it is displayed. Figure 23 shows the flow of HTTP request from the client to the Web server, which is sent to the servicing program. The program creates an HTML document to be sent to the client browser.

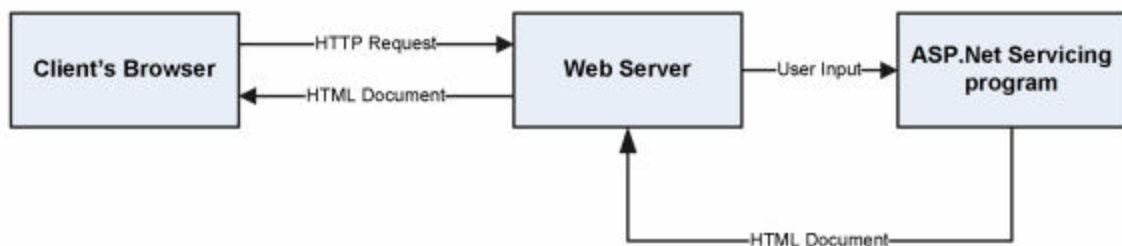


Figure 23 Compiled server programs flowchart

Popular languages for creating compiled server programs are Java, Visual Basic, and C++, but almost any language that can create executable programs can be used, provided that it supports commands used by one of the protocols that establish guidelines for communication between Web servers and servicing programs. The first such protocol, introduced in 1993, for use with HTML forms was the Common Gateway Interface (CGI); many servicing programs on Web sites still use CGI programs. However, a disadvantage of using CGI-based servicing programs is that each form submitted to a Web server starts its own copy of the servicing program on the Web server.

A busy Web server is likely to run out of memory when it services many forms simultaneously; thus, as interactive Web sites have gained popularity, Web server vendors have developed new technologies to process form inputs without starting a new copy of the servicing program for each browser input. Examples of these technologies for communicating with Web servers include Java Servlets [8] and Microsoft's ASP.NET [7]; they allow a single copy of the servicing program to service multiple users without starting multiple instances of the program.

ASP.NET has introduced many new capabilities to server-side Web programming, including a new category of elements called server controls that generate as many as 200 HTML tags and one or more JavaScript [9] functions from a single server control tag. Server controls support the processing of user events, such as clicking a mouse or entering text at either the client browser or the Web server. Server controls also encourage the separation of programming code into different files and/or areas from the HTML tags and text of a Web page, thus allowing HTML designers and programmers to work together more effectively.

Server-side scripts. Web-based applications can also use server-side scripts to create dynamic Web pages that are able to retrieve and display information from a backend database and modify data records. The processing architecture is the same as the processing architecture used for compiled server programs (Figure 21), except the Web

server processing is performed through and interpreted script rather than a compiled program.

If needed, a developer can have a single Web server process a variety of scripts written with any or all of these technologies. The Web server knows which script interpreter to invoke by taking note of the requesting script's file extension. Table 1 below demonstrates some commonly used extensions and the related technologies.

Table 1 Processing Technology for different File Extensions

File Extension	Processing technology interpreter/compiler
.asp	Microsoft Active Server Page
.aspx	Microsoft ASP.NET web page
.js	Microsoft Scripting Language "JScript" file extension
.php	PHP Script
.vbp	Visual Basic Project

Programs created through ASP.NET are not backward compatible with ASP scripts created through the original ASP server-side scripting technology [10]; upgrading older ASP scripts to ASP.NET requires substantial revision. ASP and ASP.NET programs can, however, run on the same Web server, as ASP.NET programs are distinguished with *.aspx* file extensions.

Server-side hybrid processing. Compiled server-side programs offer two main advantages: First, they are compiled and stored in a machine-readable format; so they usually run faster than scripts. Second, compiled programs are usually created in integrated development environments that provide debugging utilities. The advantage of using scripts is that their modification requires only a text editor rather than installation of an associated development environment.

Hybrid server-side programming strives to combine the advantages of compiled server-side programs and server-side scripts; a server-side script is created but not compiled. The first time a user accesses a Web page calling the script, the script is compiled into machine-readable format and stored as an executable file. With this approach, the developer works with ordinary text files and does not need to install an integrated programming development environment to modify the script. Performance is improved because the program does not need to be translated into machine language each time it runs.

In 1997, Sun Microsystems introduced Java Server Page (JSP) technology. Unlike ASP scripts, JSP source code is automatically compiled into machine-readable format the first time a user accesses it. The Web server saves the compiled JSP program, using it (rather than the source code) the next time anyone else tries to access this particular JSP. This scheme reduces both the processing performed and the time the user has to wait to view a response from the Web server. If a programmer modifies the JSP source code, the Web server notes that the source code file has been modified since the compiled version was created, compiling and saving the compiled program the next time a user access the page. In 2002, Microsoft introduced ASP.NET which is very similar to JSP with the additional flexibility that many different source languages such as VB, C#, J# can be used to implement web based applications.

Both ASP.NET and JSP are considered as Hybrid server side technologies. ASP.NET is designed to work under the Windows/Server and IIS web server environment. JSP is more portable as it works in most Operating Environments including Windows and Linux.

Choosing server-side processing. From a performance standpoint, because compiled programs execute faster than scripts, busy Web servers should use compiled server-side programs. From a development standpoint, scripts are easier to create, modify and install without having to stop the operation of the web application.

5.2. Client-Side Processing.

Client-side Web page processing is achievable through compiled programs downloaded, installed, and executed on the client workstation or by creating scripts with the HTML Web page commands interpreted by the client browser.

Downloading and running compiled programs on client workstations. When a user clicks a hyperlink on a Web page associated with a compiled client-side program, the user's browser must have the ability to run the executable program file; this program interacts with the user, sending and retrieving data from a database server as needed.

Many times, the user is asked to install certain ActiveX components to view some animations or play games. This new component *plugs in* into the existing system, thus extending the functionality of the system.

Java Applets are another example of compiled programs on client workstations. An applet is a program written in the Java programming language that can be included in an HTML page, much in the same way an image is included in a page. When we use a Java technology-enabled browser to view a page that contains an applet, the applet's code is transferred to our system and executed by the browser

Client-side scripts. In a client-side script, source code written in such languages as JavaScript and VBScript is embedded in an HTML document, along with the static HTML text; it is placed within delimiter tags to indicate to the user's browser that the text is code that must be interpreted. If the user's browser is able to recognize and interpret the code, it is processed. If the browser is unable to recognize and interpret the code, it is displayed as text on the Web page.

Although basic client-side scripts cannot be used by a Web page to interact with a remote database, they are often used to validate user inputs entered on HTML forms submitted for processing by a server-side program; for example, a script running on a

client workstation might check the inputs users submit to a Web page to make sure they entered all required data and appropriate data values. This approach avoids transmitting inputs to the Web server that are incomplete or include errors, while offloading error checking and handling from the Web server program to the client workstation.

Client-side scripts can also be used to create advanced Web page features, including: animations, calculations, playing sound and video, and image maps allowing users to move their cursors over an image and click to access different Web page links.

JavaScript is the most commonly used client-side scripting language and is supported by most browsers.

Use of a client-side scripting language depends on the user's operating system, browser platforms, and developer expertise. If the Web pages in question are to be accessed by a variety of users over the Internet, JavaScript is probably better than VBScript, as JavaScript is the only scripting language able to run on nearly all browsers. If the Web pages are to be accessed on an intranet and if the organization has standardized on Microsoft's browser and Web server, VBScript is a satisfactory scripting language for creating client-side scripts.

6. Web Based Application Development

The Web is built on the HyperText Transfer Protocol. HTTP is a client/server request/reply protocol that is *stateless*. That is, the protocol does not make any association between one transaction and another; e.g.: time since the last transaction, type or client involved in the last transaction, what data was exchanged between the client and the server. As far as HTTP is concerned, each transaction is a discrete event. But this is not what we want in a shopping cart application because we need to preserve the user's shopping selection as they proceed with their purchase, in addition it is useful to have the access to their past purchase history and personal preferences.

Carrying information from one page to another can be achieved by several ways, such as Cookies, Session variables, Post variables, etc.

A cookie is a small file that has a maximum age, a domain and path of applicability, and a security specification. Any time a server sends a response to a client, it may include one or more Set-Cookie headers. When a client receives a Set-cookie header, it stores the content of the header and the cookie, for later use. In our application, every time the client selects an item to put in the shopping cart, the server can send a Set-Cookie whose content is the ID of the item, and whose domain and path of applicability are the URL of the order/payment page. Then, when the user goes to order and pay, the client will send the Cookie headers for each of the selected items. Upon receiving this request, the server can parse the supplied cookies and charge the user appropriately for the selected items. Cookies may also be used to identify the users.

However, cookies are very insecure to use since they are transmitted as plain text and the server has no control over how cookies are stored in at the client's side. Another approach is based on a notion of session ID. These notions provide means for the server to track the requests of a client through a "session", but unlike cookies, which are stored on the client, Session variables are stored on the Server. A session starts when a user logs in and ends when they log off from the website.

The Session object is used to store information about, or change settings for a user session. Variables stored in the Session object hold information about one single user, and are available to all pages in one application. Common information stored in session variables are name, id, and preferences. The server creates a new Session object for each new user, and destroys the Session object when the session expires.

In this project, the concept of session variables will be used for maintaining state information.

7. Database Connectivity

In e-commerce applications it is very typical for the Web server to contact the database to get information as needed. ASP.NET uses a technology called ActiveX Data Objects.NET (ADO.NET) to connect to the database.

7.1 ADO.NET

Classic ASP pages used ActiveX Data Objects (ADO) to access and modify databases. ADO is a programming interface used to access data. This method was efficient and fairly easy for developers to learn and implement. However, ADO suffered from a dated model for data access with many limitations, such as the inability to transmit data so it is easily and universally accessible. Coupled with the move from standard SQL databases to more distributed types of data (such as XML), Microsoft introduced ADO.NET.

Although ADO.NET is known as the next evolution of ADO, it is very different from its predecessor. Whereas ADO was connection-based, ADO.NET relies on short, XML message-based interactions with data sources. This makes ADO.NET much more efficient for Internet-based applications.

A fundamental change from ADO to ADO.NET was the adoption of XML for data exchanges. XML is a text-based markup language, similar to HTML that presents an efficient way to represent data. This allows ADO.NET to reach and exchange. It also gives ADO.NET much better performance because XML data is easily converted to and from any type of data.

Another major change is the way ADO.NET interacts with databases. ADO requires “locking” of database resources and lengthy connections for its applications, but ADO.NET does not; it uses disconnected data sets, which eliminates lengthy connections

and database locks. This makes ADO.NET much more scalable because users are not in contention for database resources.

In ADO.NET there are two core objects that allow us to work with data initially: the `DataReader` and the `DataSet`. In any .NET data access page, before we connect to a database, we first have to import all the necessary namespaces that will allow us to work with the objects required. Namespace in .NET is a set of classes that can be used while creating an application. The .NET Framework has about 3,500 classes which can be accessed through a namespace. The application will be using a technology known as Open DataBase Connectivity (ODBC) to access the database; therefore we must first import necessary namespaces. Below is a sample namespace declaration used by .NET.

```
<%@ Import Namespace="System" %>
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.Odbc" %>
```

After all the necessary namespaces are imported, a connection to the database is made.

```
OdbcConnetion odbcCon = new OdbcConnection ("DRIVER = {MySQL ODBC 3.51
Driver}; SERVER=localhost; DATABASE=project;
UID=root; PASSWORD=pwd" );

odbcCon.Open( );
```

The above statement creates a connection to the database with an `OdbcConnection` object. This object tells ASP.NET where to go to get the data it needs. Since the data is stored in the same computer as the application, the `SERVER` is given as *localhost*. Next we open the connection object. Listed below are the common connection object methods we could work with:

- ***Open*** - Opens the connection to our database
- ***Close*** - Closes the database connection
- ***Dispose*** - Releases the resources on the connection object. Used to force garbage collecting, ensuring no resources are being held after our connection is used.
- ***State*** - Tells you what type of connection state your object is in, often used to check whether the connection is still using any resources.

Once the connection is made, in order to access the data in a database, ADO.NET relies on two components: DataSet and Data Provider [20]. These components are explained below.

DataSet

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 DataSet resides in memory and the data in it can be manipulated and updated independent of the database. If necessary, changes made to the dataset can be applied to the central database. The data in DataSet can be loaded from any valid data source such as a text file, an XML database, Microsoft SQL server database, an Oracle database or MySQL database.

Data Provider

The Data Provider is responsible for providing and maintaining the connection to the database. A DataProvider is a set of related components that work together to provide data in an efficient and performance driven manner. Each DataProvider 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 DataReader object which provides a read only, connected recordset
- The DataAdapter object which populates a disconnected DataSet with data and performs the update.

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.

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. Command objects provide three methods that are used to execute commands on the database:

ExecuteNonQuery: Executes commands that have no return values such as INSERT, UPDATE or DELETE.

ExecuteScalar: Returns a single value from a database query

ExecuteReader: Returns a result set by way of a DataReader object

The DataReader Object

The DataReader object provides a 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 one does not need to keep the data cached in memory. Because only one row is in memory at a time, the DataReader provides the lowest overhead in

terms of system performance but requires the exclusive use of an open Connection object for the lifetime of the DataReader.

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 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

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.

ADO.NET follows the below process, Figure 24, to connect to the database and retrieve data to the application [21].

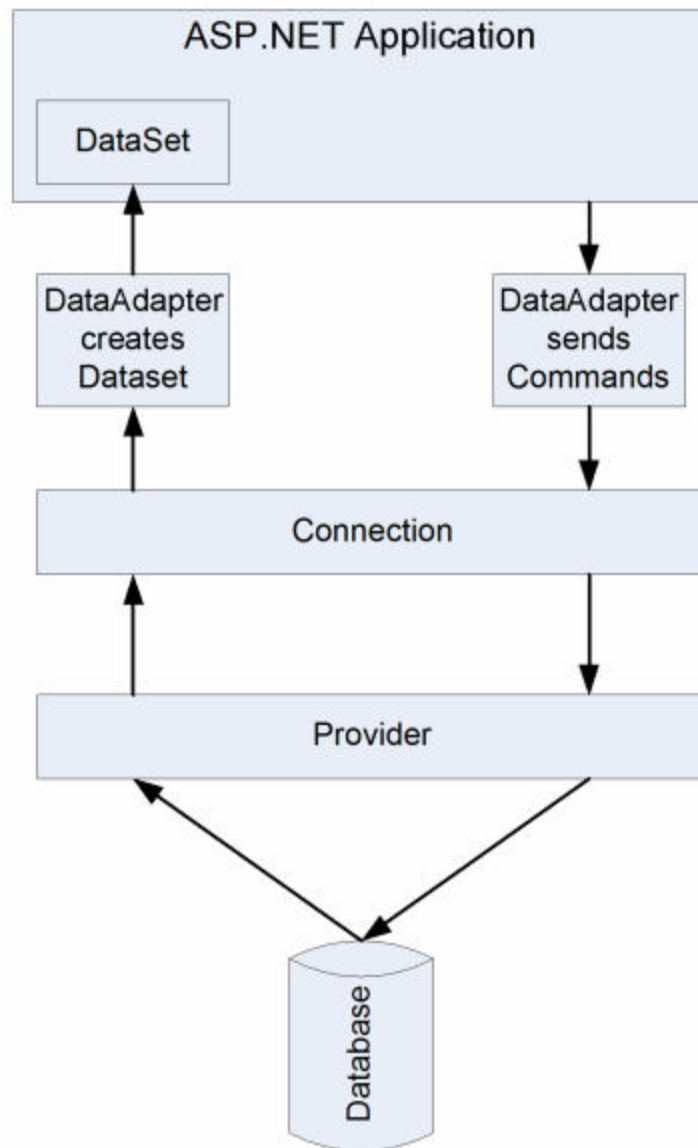


Figure 24 ADO.NET Architecture

- When an ASP.NET application needs to access the database, it submits an appropriate request to ADO.NET through a DataAdapter object, which in turn sends a command to the Connection object.
- The Connection object establishes a connection to the database and submits the request sent by DataAdapter.
- The Connection object connects to the database through a Provider such as ODBC.NET. The Provider acts as a translator between the Connection object and the database. It translates the request for data to database's language and brings back the data, if needed.
- The Provider sends the data back to the DataAdapter through the Connection object and DataAdapter places the data in a DataSet object residing in application's memory.

Instead of storing data in a DataSet, a DataReader can be used to retrieve data from the database. Results are returned in a resultset which is stored in the network buffer on the client until a request is made to **Read** method of the DataReader. Using the DataReader can increase the application performance by retrieving as soon as the data is available, rather than waiting for the entire results of the query to be returned [22].

A DataSet can be used to interact with data dynamically such as binding to a Web Form, cache locally in the application, provide hierarchical XML view of the data, etc. If such functionalities are not required by the application, a DataReader can be used to improve the performance of the application. By using a DataReader, the memory can be saved that is used by the DataSet, as well as the processing required to Fill the contents of a DataSet.

When a DataReader is used, a DataAdapter is not required to send the data to the application. In this project, DataReader is used to read the data and Command object called ExecuteNonQuery is used to write into the database.

7.2 Connecting ASP.NET application to a Database

The steps required to connect our ASP.NET application to the MySQL database and access the data are given below:

1. Import the required namespaces.

```
using System;
using System.Data;
using System.Data.Odbc;
```

2. Create a connection object.

```
string myConnectionString;
myConnectionString = "DRIVER = {MySQL ODBC 3.51 Driver}; SERVER =
                      localhost; DATABASE = project; UID = root;
                      PASSWORD = ''"
OdbcConnection odbcCon = new OdbcConnection(myConnectionString)
```

3. Create a SQL query

```
string str;
str="Select * from Customer where UserID='admin';
```

4. Create a Command object to run the SQL query

```
odbcCmd=new OdbcCommand(str,odbcCon);
```

5. DataReader to read the result

```
OdbcDataReader odbcReader;
String text, text2;
while (odbcReader.Read())
{
    text = odbcReader["UserID"].ToString();
    text2 = odbcReader["FirstName"].ToString();
}
```

6. Close odbcReader and odbcConnection

```
odbcReader.Close();
odbcCon.Close();
```

The data can now be used as desired by the application.

8. The Shopping Cart Application

The objective of this application is to provide the user an online website where they can buy books from the comfort of their home. A shopping cart is used for the purpose. The user can select the desired books, place them in the shopping cart and purchase them using a Credit Card. The user's order will be shipped according to the type of shipping selected at the time of placing the order.

Website consists of the following web pages:

1. AddBook.aspx
2. BookDetails.aspx
3. BookReview.aspx
4. Books.aspx
5. ChangePassword.aspx
6. CheckOut.aspx
7. FinalOrder.aspx
8. Footer.ascx
9. ForgotPassword.aspx
10. Login.aspx
11. LogOff.aspx
12. Menu.ascx
13. Order.aspx
14. PurchaseHistory.aspx
15. Registration.aspx
16. Search.aspx
17. ShoppingCart.aspx
18. UserDetails.aspx

Below figures show some screenshots taken from running the application. All the functionalities are explained accordingly.

When the user types the web address in the browser, the main page of the application is displayed which has the list of the top ten popular books available in the store, as shown in Figure 25.

[Books](#)
[Search](#)
[Shopping Cart](#)
[User Details](#)
[Registration](#)

[Login](#)

Book Details

ISBN	Book Name	Author	Number of Books	Price (\$)		
0596006993	Programming C#	Jesse Liberty	24	44.99	Add to cart	Book Review
0596001711	Programming ASP.NET	Jesse Liberty, Dan Hurwitz	25	14.49	Add to cart	Book Review
1583227131	A Man without a Country	Kurt Vonnegut	25	24.47	Add to cart	Book Review
0672326663	Programming in C	Stephen Kochan	25	34.45	Add to cart	Book Review
0764549979	Beginning Programming for Dummies	Wallace Wang	25	24.99	Add to cart	Book Review
1594200637	On Beauty	Zadie Smith	25	26.95	Add to cart	Book Review

Figure 25 Book Details

The information about books is stored in “Books” table. The user can know the ISBN of the book, book title, author of the book, number of copies available at the store, price of the book. A link to add the book to the shopping cart and also a link to write a review for the book are also provided. The user does not have to login to add a book to the cart or to read/write a review.

8.1. Search for Books

Books can be searched based on the ISBN, Title or the Author of the book. When searching for books by author “Jesse Liberty”, two books are displayed as shown in Figure 26.

[Login](#)

Books	Search	Shopping Cart	User Details	Registration
-------	--------	---------------	--------------	--------------

Search

ISBN:

Book Name

Author:

2 book(s) found

ISBN	Book Name	Author	Number of books	Price of the book	
0596001711	Programming ASP.NET	Jesse Liberty, Dan Hurwitz	25	14.49	Add to cart
0596006993	Programming C#	Jesse Liberty	24	44.99	Add to cart

Figure 26 Search for books

8.2. Registration

A new user can register on the site by clicking on the registration button on the menu at the top of the page, as shown in Figure 27.

Books	Search	Shopping Cart	User Details	Registration
--------------	---------------	----------------------	---------------------	---------------------

Registration

[Login](#)

User ID*

Password*

Confirm Password*

First Name*

Last Name

Address*

City*

Zip* **Required**

State* 

Email*

Daytime Phone Number

Copyright © Swapna Kodali, 2005

Figure 27 New user registration

The “*” beside the label indicates the required field for successful registration on the site. If the value is not entered, an appropriate message is displayed. If a user with same UserID already exists, the message is displayed. Clicking on Reset will clear all the fields and Submit will submit the information for registration. Upon successful completion, the user is directed to the Books page.

8.3. User Details

On clicking “User Details”, the detailed profile information of the user who is currently logged in are displayed as shown in Figure 28.

The screenshot displays a web application interface for user details. At the top right, the user 'skodali' is logged in, with a 'Log off' link. A navigation bar contains five buttons: 'Books', 'Search', 'Shopping Cart', 'User Details' (which is highlighted), and 'Registration'. The main heading is 'User Details'. Below this, the 'User ID' is 'skodali'. The form includes fields for 'First Name' (Swapna), 'Last Name' (Kodali), 'Email' (skodali@iusb.edu), 'Address' (1986 Ridgegate Lane), 'City' (Simi Valley), 'State' (CA), 'Zip' (93065), and 'Phone' (1234567890). There are 'Update' and 'Cancel' buttons, a 'Change Password' button, and a 'Purchase History' button.

Books	Search	Shopping Cart	User Details	Registration
-------	--------	---------------	--------------	--------------

User Details

User ID: skodali

First Name

Last Name

Email:

Address:

City:

State:

Zip

Phone:

Figure 28 User details

Here the users can change their profile except for the UserID and these details will be reflected in the database only when the Update button is clicked. If the user has placed any order previously, those details can be viewed by clicking on the Purchase History button.

8.4. Shopping Cart

When “Add to Cart” is clicked for any book, it is added to the shopping cart illustrated in Figure 29. If that particular book is already present in the shopping cart, the quantity is increased by 1 and the price is changed accordingly; if not, a new entry is made into the table. All the information in the shopping cart is stored in “shopping_cart_items” table. Adding a book into the shopping cart does not decrease the quantity of books in the Books table. It is decreased only after an order is placed for the book. So, placing the book in the shopping cart does not guarantee the availability of the book at the time of placing the order.

skodali, [Log off](#)

Books	Search	Shopping Cart	User Details	Registration
-------	--------	---------------	--------------	--------------

Shopping Cart

ISBN	Book Name	Quantity	Price
0596001711	Programming ASP.NET	2	28.98
1583227131	A Man without a Country	1	24.47

[Place an Order](#)

Copyright © Swapna Kodali, 2005

Figure 29 Shopping cart

8.5. Place an Order

When “Place an Order” button is clicked which is located on the bottom of the shopping cart, the application will ask the user to login if he has not already done so.

Order Details

Shopping Cart

Book Name	Quantity	Price
Programming ASP.NET	2	28.98
A Man without a Country	1	24.47

Credit Card Type

Master Card

Credit Card Holder Name

Swapna Kodali

Credit Card Number

1234123412341234

Expiry date

12/06

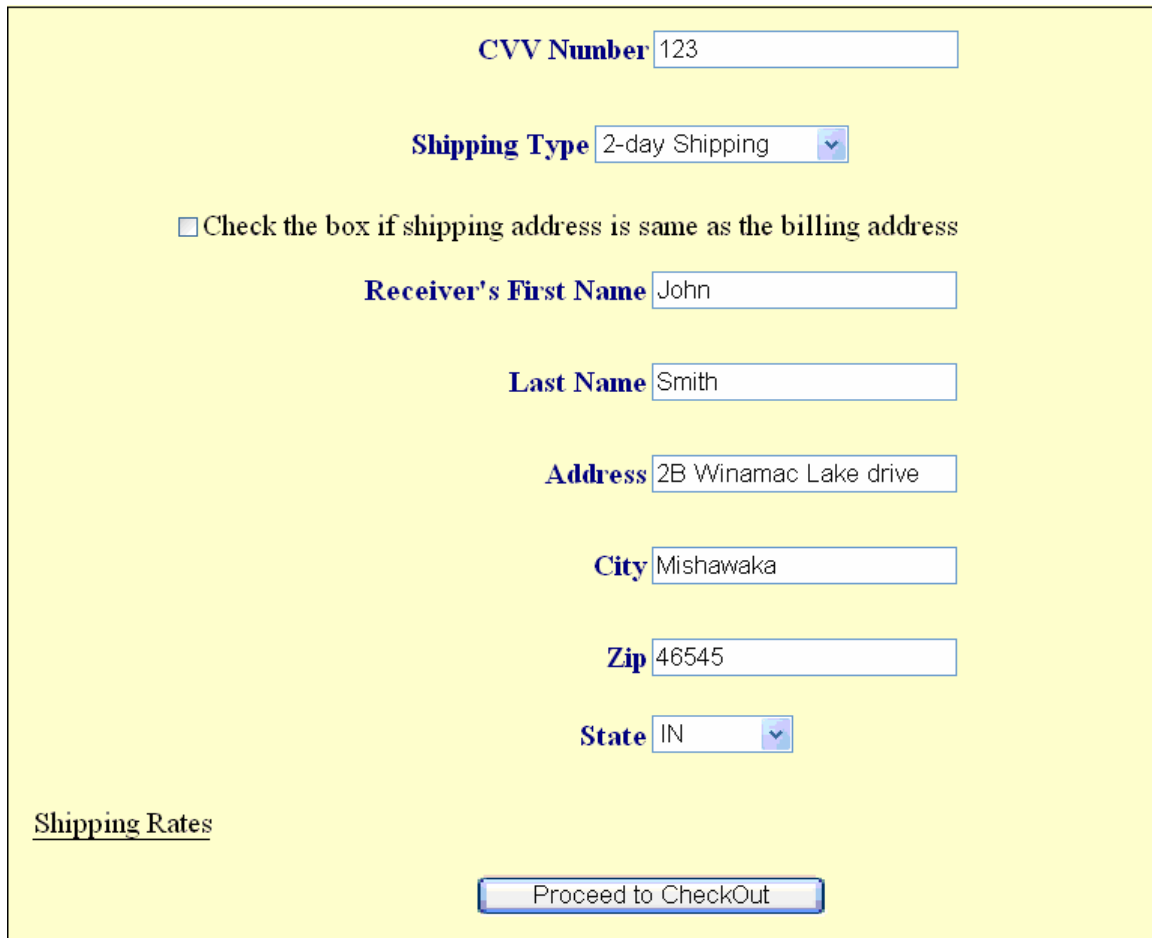
CVV Number

123

Figure 30 Order Details

If the user is placing an order with the web site for the first time, they will be asked to enter the credit card details as shown in the above figure; if not, only the Card Verification Value (CVV) number of the credit card is asked for verification, as shown in Figure 30.

At this point, the user can check the shipping address box if shipping address is same as billing address, otherwise the user has to enter the new shipping address as shown in Figure 31.



The form is titled "Shipping details" and is set against a light yellow background. It contains several input fields and a checkbox. At the top, there is a "CVV Number" field with the value "123". Below that is a "Shipping Type" dropdown menu showing "2-day Shipping". A checkbox is labeled "Check the box if shipping address is same as the billing address". Following the checkbox are fields for "Receiver's First Name" (John), "Last Name" (Smith), "Address" (2B Winamac Lake drive), "City" (Mishawaka), "Zip" (46545), and "State" (IN). At the bottom left, there is a link "Shipping Rates" with a red underline. At the bottom center is a "Proceed to CheckOut" button.

CVV Number	123
Shipping Type	2-day Shipping
<input type="checkbox"/> Check the box if shipping address is same as the billing address	
Receiver's First Name	John
Last Name	Smith
Address	2B Winamac Lake drive
City	Mishawaka
Zip	46545
State	IN
Shipping Rates	
Proceed to CheckOut	

Figure 31Shipping details

If the check box provided is checked, the shipping address is obtained from the *Customer* table. The user also has to select the desired type of shipping for the order. When all the information is entered, the user can "Proceed to the Checkout".

8.6. Check Out

Before placing the final order, the user is shown the total price of the order, which includes total price of books selected, shipping rate and state tax as illustrated in Figure 32. If the user is not satisfied with the order, the order can be cancelled at that point. The information in the shopping cart remains intact, so the user can go back to it and make any changes if necessary. When the “Place Order” button is clicked, the order is placed and the following screen appears which informs the user about the approximate number of days in which the order will be delivered.

skodali, [Log off](#)

Books	Search	Shopping Cart	User Details	Registration
-------	--------	---------------	--------------	--------------

Shopping Cart

Book Name	Quantity	Price
Programming ASP.NET	2	28.98
A Man without a Country	1	24.47

Shipping Rate	7.49
Tax	6%
Total Price	64.6

[Place Order](#)

Figure 32 Check out

Once the order is placed, the quantity of the books is reduced in the *Books* table. The shopping cart for the user cleared and an appropriate message is displayed, as shown in Figure 33.

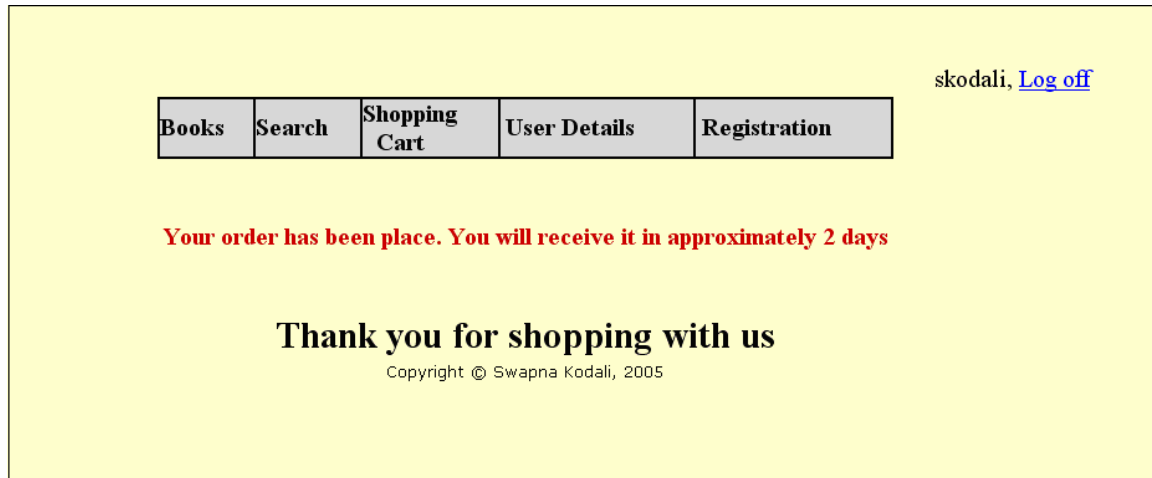


Figure 33 Order confirmation

The inventory is updated as shown in Figure 34 after the order is placed.

ISBN	Book Name	Author	Number of Books	Price (\$)		
0596006993	Programming C#	Jesse Liberty	24	44.99	Add to cart	Book Review
0596001711	Programming ASP.NET	Jesse Liberty, Dan Hurwitz	23	14.49	Add to cart	Book Review
1583227131	A Man without a Country	Kurt Vonnegut	24	24.47	Add to cart	Book Review
0672326663	Programnning in C	Stephen Kochan	25	34.45	Add to cart	Book Review
0764549979	Beginning Programming for Dummies	Wallace Wang	25	24.99	Add to cart	Book Review
1594200637	On Beauty	Zadie Smith	25	26.95	Add to cart	Book Review

Figure 34 Updated inventory after order placement

8.7. Purchase History

Figure 35 details the purchase history of the user “skodali”. Purchase history can be reached by clicking on the “Purchase History” tab on “User Details” screen as shown in the Figure 28.

skodali, [Log off](#)

Books	Search	Shopping Cart	User Details	Registration
-------	--------	---------------	--------------	--------------

Purchase History

Book Name	Quantity	Price	Purchase Date
Programming ASP.NET	2	7.49	12/9/2005 12:00:00 AM
A Man without a Country	1	7.49	12/9/2005 12:00:00 AM

Copyright © Swapna Kodali, 2005

Figure 35 Purchase history

When viewing the purchase history, the user can view the details of each book by clicking on the book name. The details are displayed as shown in Figure 36.

admin, [Log off](#)

Books	Search	Shopping Cart	User Details	Registration
-------	--------	---------------	--------------	--------------

Book Details

Book Name:

Author:

Cost:

ISBN:

[Books](#)

Copyright © Swapna Kodali, 2005

Figure 36 Book details

Book information can only be changed by the Administrator of the site. All other users can only view the details of the books. The administrator of the site can also “Add Book” or “Remove Book” to/from the *Books* table. Figure 37 allows a book modification form accessible to the administrator.

The screenshot shows a web interface for an administrator. At the top right, it says "admin, [Log off](#)". Below this is a navigation bar with five buttons: "Books", "Search", "Shopping Cart", "User Details", and "Registration". The "Books" button is highlighted. In the center, the text "Add New Book" is displayed in a large, bold, red font. Below this, there is a link "[Add Book](#)". Further down, the label "ISBN" is followed by a text input field. Below the input field is a button labeled "Check/Add". At the bottom of the page, the copyright notice "Copyright © Swapna Kodali, 2005" is visible.

Figure 37 Administrator - Modify books

In order to add a book, the administrator will enter the ISBN of the book. If the ISBN is already present in the Books table, the administrator is asked to enter the quantity of the books.

admin, [Log off](#)

Books	Search	Shopping Cart	User Details	Registration
-------	--------	---------------	--------------	--------------

Add New Book

[Add Book](#)

ISBN

Book Title

Author

Cost

Quantity

Figure 38 Details about new book

If the entered ISBN book is not present in the Inventory, the administrator is asked to enter the details about the book as shown in Figure 38 to add the book to the inventory.

Figure 39 shows the updated inventory after the book details in Figure 38 are entered.

admin, Log off				
Books	Search	Shopping Cart	User Details	Registration
Add Book				
Book Details				
ISBN	Book Name	Author	Number of Books	Price (\$)
0596006993	Programming C#	Jesse Liberty	24	44.99
0596001711	Programming ASP.NET	Jesse Liberty, Dan Hurwitz	21	14.49
1583227131	A Man without a Country	Kurt Vonnegut	23	24.47
0672326663	Programming in C	Stephen Kochan	25	34.45
0764549979	Beginning Programming for Dummies	Wallace Wang	25	24.99
1594200637	On Beauty	Zadie Smith	25	26.95
0957921861	Build your own Website using ASP.NET	Zak Ruvalcaba	10	29.67

Figure 39 Updated Inventory

When a user logs off the website, the items in their Shopping Cart are cleared.

When the user enters a password during the registration, it is encrypted before it is stored in the database. It is a one-way encryption and the original form cannot be retrieved again. Similar encryption method is used for Credit Card Number.

Suppose there are only three “Programming C#” books available in the store. One user adds all of them to the shopping cart, and by the time he chooses other books and places the order, another user has already placed an order for two of those books. In that case, the first user comes to know about this at the time of placing the order and he is directed to the shopping cart to make the appropriate changes.

As explained earlier, the user need not be logged in to add books to the shopping cart. When the user adds books without logging in, a GUID (Globally Unique Identifier) is obtained from the system and stored in Session["loginid"] variable. This GUID is stored in the shopping_cart_items table along with the selected books by the user. If the user logs in, the GUID in the table is replaced with the actual UserID of the user.

Session variables are used to transfer data from one page to another. As soon as the user closes the window, the session variables are cleared.

8.8. Transactions in the Application

A transaction is a group of database commands that are treated as a single unit. Transaction must pass what is known as the ACID test:

Atomic: All operations in the transaction are executed properly or none. In other words, they make up a single unit of work. For example, if a customer moves and a transaction is used to reflect that change in the database, all parts of the address (street, city, state, etc) must be changed as an atomic action, rather than changing street, then city, then state, and so on.

Consistent: The execution of a single transaction preserves the consistency of the database. All the relationships between data in a database are maintained correctly. For example, if customer information uses a tax rate from a state tax table, the state entered for the customer must exist in the state tax table.

Isolation: Each transaction is unaware of the other transactions occurring concurrently. Changes made by other clients cannot affect the current changes. For example, if two data entry operators try to make a change to the same customer at the same time, one of two things occurs: either one operator's changes are accepted and the other is notified that the changes were not made, or both operators are notified that their changes were not made. In either case, the customer data is not left in an indeterminate state.

Durability: Changes the transaction has performed persist in the database. Once a change is made, it is permanent. If a system error or power failure occurs before a set of commands is complete, those commands are undone and the data is restored to its original state once the system begins running again.

Transaction processing is particularly important for Web applications that use data access, since Web applications are distributed among many different clients. In a Web application, databases are a shared resource, and having many different clients distributed over a wide area can present these key problems:

- Contention for resources. Several clients might try to change the same record at the same time. This problem gets worse the more clients you have.
- Unexpected failures. The Internet is not the most reliable network, even if your Web application and Web server are 100 percent reliable. Clients can be unexpectedly disconnected by their service providers, by their modems, or by power failures.
- Web application life cycle. Web applications do not follow the same life cycle as Windows applications—Web forms live for only an instant, and a client can leave your application at any point by simply typing a new address in their browser.

Transaction processing follows these steps:

1. Begin a transaction.
2. Process database commands.
3. Check for errors.
4. If errors occurred, restore the database to its state at the beginning of the transaction. If no errors occurred, commit the transaction to the database.

Suppose two users try to add the same book to the shopping cart and try to place an order at the exact same time. An update should be done to the Books table after the order is placed, but if only the latest transaction is noted down, the book quantity will differ in the real world. This situation has to be handled as in a “Transaction”. As detailed earlier, a transaction is an operation or set of operations that succeeds or fails as a logical unit. That is, either both the updates are not done, or both the updates are done consecutively.

Transactions are normally managed by declaring boundaries around a set of operations. Operations that execute in the context of the transaction boundary then succeed or fail as a unit. For ASP.NET, the transaction boundary is the execution of a single request to a page, which might contain nested components that participate in the same transaction. While the page is executing, if an operation on the page itself or a nested component in the same transaction fails, it can call **ContextUtil.SetAbort**. This is then picked up by the current transaction context, the entire transaction fails, and any operations that were already completed are undone. If nothing fails, the transaction is committed.

ASP.NET support for transactions consists of the ability to allow pages to participate in ongoing Microsoft .NET Framework transactions. Transaction support is exposed via an **@Transaction** directive that indicates the desired level of support:

```
<% @ Transaction="Required" %>
```

Table 2 defines the supported transaction attributes. The absence of a transaction directive is the same as an explicit directive to "Disabled".

Table 2 Transaction attributes

Attribute	Description
Required	The page requires a transaction. It runs in the context of an existing transaction, if one exists. If not, it starts one.
RequiresNew	The page requires a transaction and a new transaction is started for each request.
Supported	The page runs in the context of an existing transaction, if one exists. If not, it runs without a transaction.
NotSupported	The page does not run within the scope of transactions. When a request is processed, its object context is created without a transaction, regardless of whether there is an active transaction.

A transaction can be explicitly committed or aborted using static methods of the **System.EnterpriseServices.ContextUtil** class. You can explicitly call the **SetComplete** or **SetAbort** method to commit or abort an ongoing transaction.

A transaction will commit or abort at the end of the page's lifetime depending on whether **SetComplete** or **SetAbort** was called last, provided there is no other object to join the same transaction.

9. Limitations and Future Development

There are some limitations for the current system to which solutions can be provided as a future development:

1. The system is not configured for multi-users at this time. The concept of *transaction* can be used to achieve this.
2. The Website is not accessible to everyone. It can be deployed on a web server so that everybody who is connected to the Internet can use it.
3. Credit Card validation is not done. Third party proprietary software can be used for validation check.

As for other future developments, the following can be done:

1. The Administrator of the web site can be given more functionalities, like looking at a specific customer's profile, the books that have to be reordered, etc.
2. Multiple Shopping carts can be allowed.

10. Conclusion

The Internet has become a major resource in modern business, thus electronic shopping has gained significance not only from the entrepreneur's but also from the customer's point of view. For the entrepreneur, electronic shopping generates new business opportunities and for the customer, it makes comparative shopping possible. As per a survey, most consumers of online stores are impulsive and usually make a decision to stay on a site within the first few seconds. "Website design is like a shop interior. If the shop looks poor or like hundreds of other shops the customer is most likely to skip to the other site"[16]. Hence we have designed the project to provide the user with easy navigation, retrieval of data and necessary feedback as much as possible.

In this project, the user is provided with an e-commerce web site that can be used to buy books online. To implement this as a web application we used ASP.NET as the Technology. ASP.NET has several advantages such as enhanced performance,

scalability, built-in security and simplicity. To build any web application using ASP.NET we need a programming language such as C#, VB.NET, J# and so on. C# was the language used to build this application. For the client browser to connect to the ASP.NET engine we used Microsoft's Internet Information Services (IIS) as the Web Server. ASP.NET uses ADO.NET to interact with the database as it provides in-memory caching that eliminates the need to contact the database server frequently and it can easily deploy and maintain an ASP.NET application. MySQL was used as back-end database since it is one of the most popular open source databases, and it provides fast data access, easy installation and simplicity.

A good shopping cart design must be accompanied with user-friendly shopping cart application logic. It should be convenient for the customer to view the contents of their cart and to be able to remove or add items to their cart. The shopping cart application described in this project provides a number of features that are designed to make the customer more comfortable.

This project helps in understanding the creation of an interactive web page and the technologies used to implement it. The design of the project which includes Data Model and Process Model illustrates how the database is built with different tables, how the data is accessed and processed from the tables. The building of the project has given me a precise knowledge about how ASP.NET is used to develop a website, how it connects to the database to access the data and how the data and web pages are modified to provide the user with a shopping cart application.