GRT INSTITUTE OF ENGINEERING AND TECHNOLOGY, TIRUTTANI - 631209

Approved by AICTE, New Delhi Affiliated to Anna University, Chennai

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PHASE 3

PROJECT TITLE

E-COMMERCE APLLICATION ON IBM CLOUD FOUNDARY

COLLEGE CODE : 1103

MAHAMMAD ALI SK

3rd yr, 5th sem

Reg no. : 110321104047

crazymahammad@gmail.com

In this phase we will begin building our project. In this technology projects we will begin building your project using IBM Cloud Foundry.

Abstract

In today's digital age, ecommerce apps have transformed the way we shop and interact with businesses. From personalized recommendations to virtual try-on experiences, these apps offer convenience, accessibility, and a seamless shopping experience. In this blog post, we will explore 10 innovative ecommerce app ideas that have the potential to revolutionize online shopping. Let's dive in!

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.

2. 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. 4 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 Re lational model.

3. 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.

3.1. 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).

3.2 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.

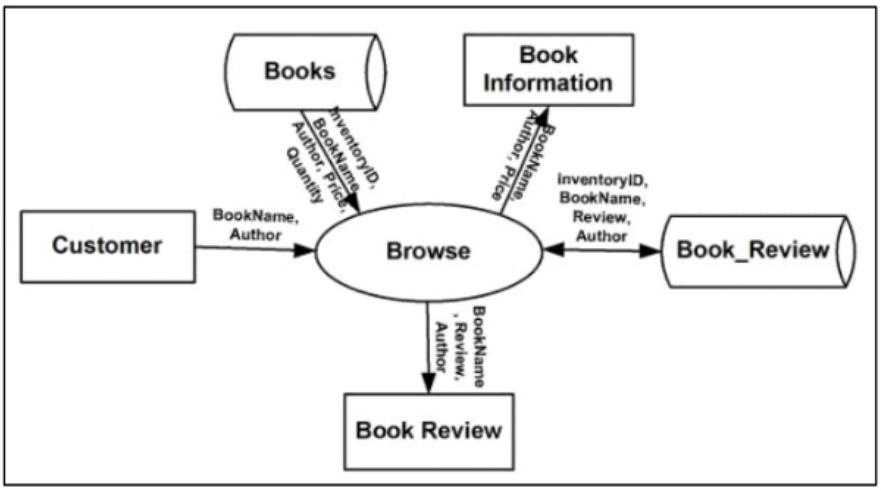
2. Ellipses representing processes, which take data as input, validate and

process it and output it.

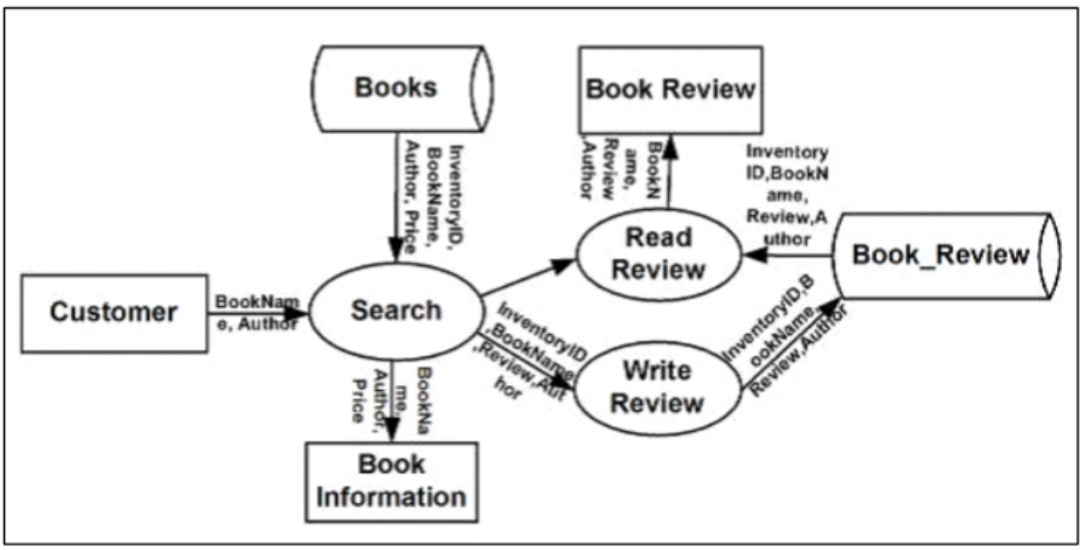
3. Arrows representing the data flows, which can either, be electronic data or physical items.

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.

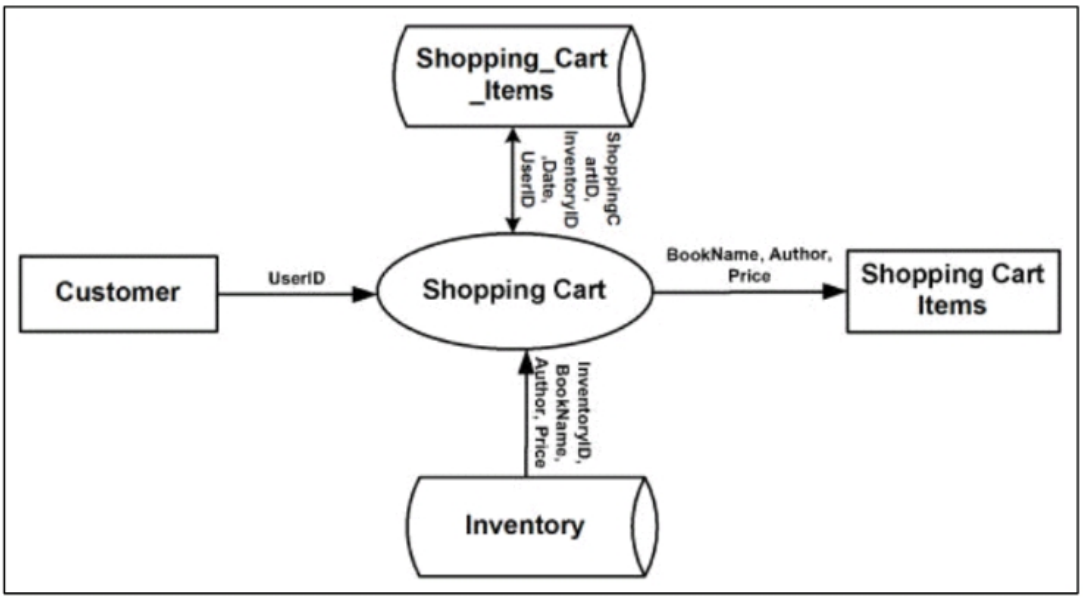
Customer-Browse Context DFD

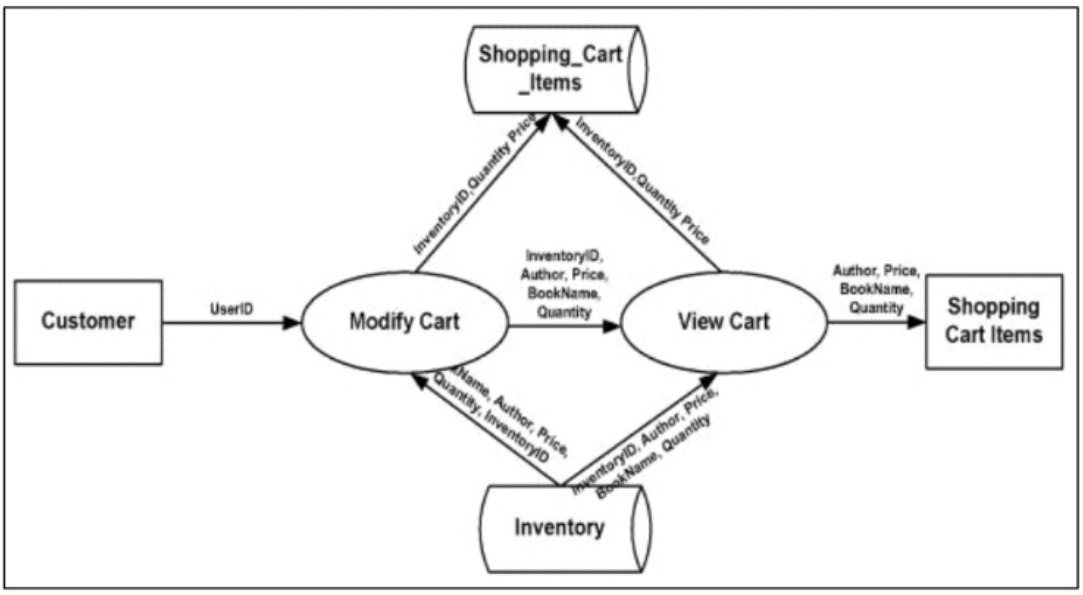


Customer-Browse Detailed DFD

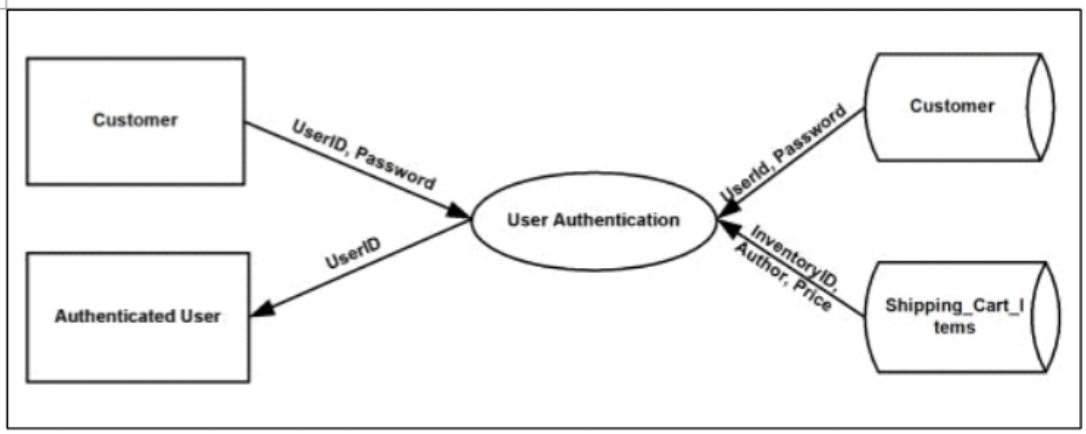


Customer - ShoppingCart Context DFD

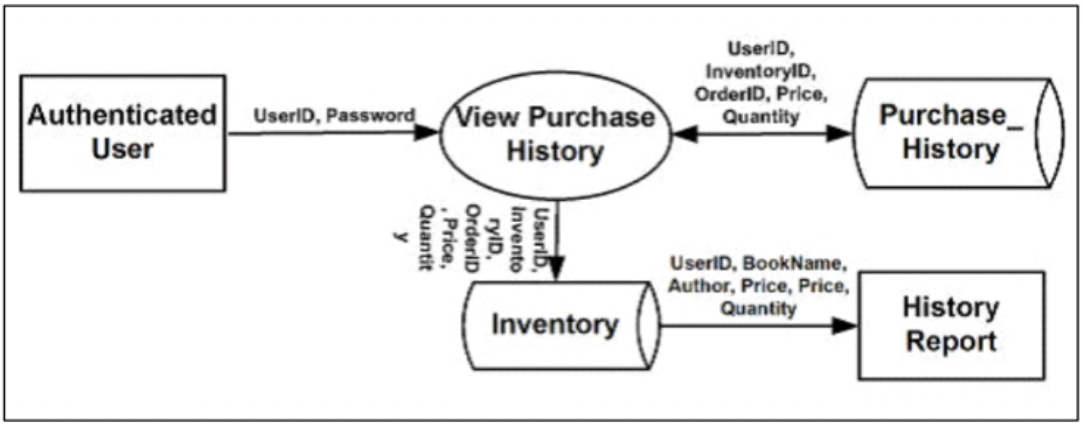




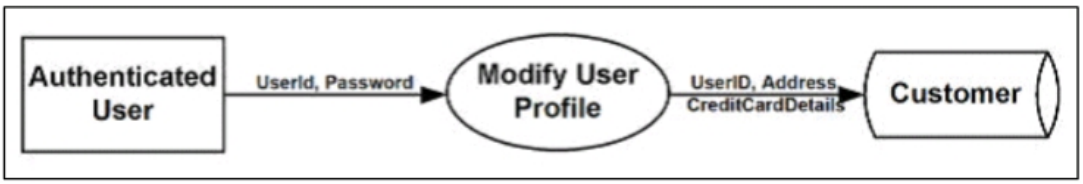
Customer-Authentication Context DFD



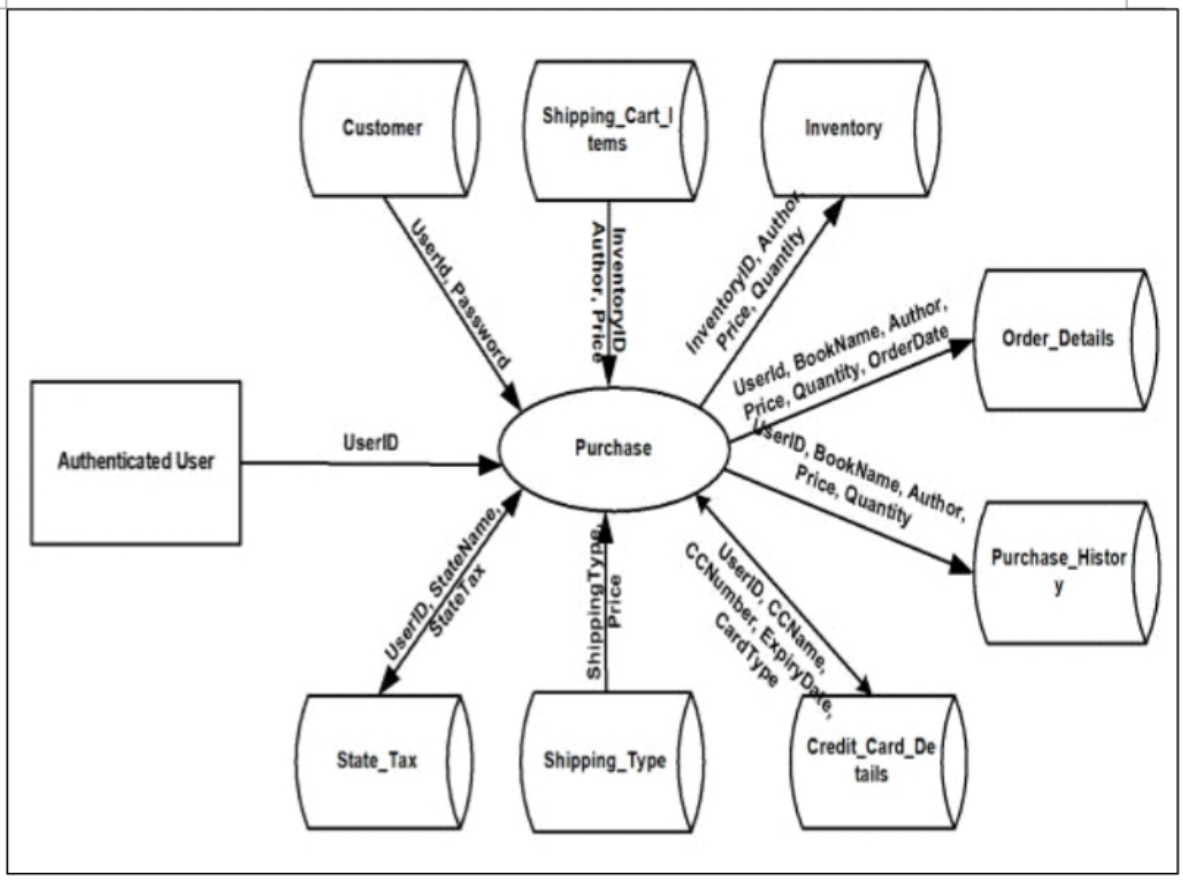
Customer-Authentication-PurchaseHistory DFD



Customer-Authentication-UserProfile DFD

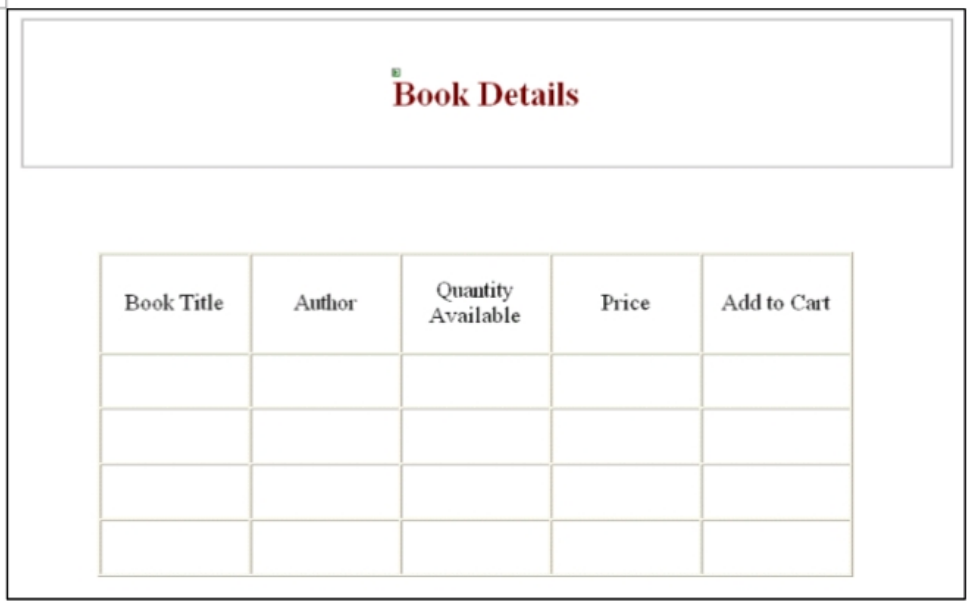


Authenticated User-Purchase Context DFD



4. 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.



5. 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

6.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 isused 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. Forexample, 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*

7. 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

9. 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 la st 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-cookieheader, 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.

10 . 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.

10.1 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 whe re 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

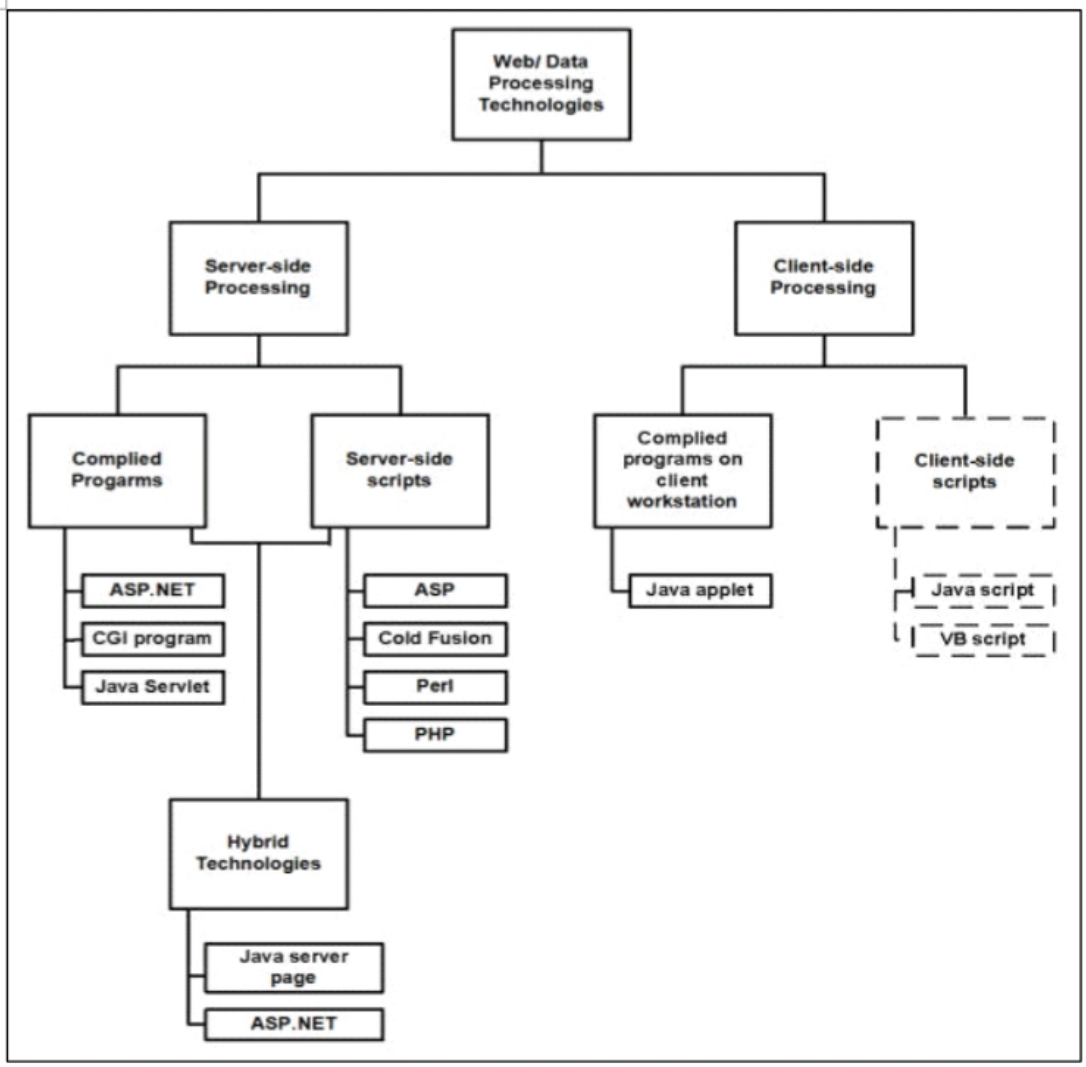
The DataAdapter is the class at the core of ADO .NET's disconnected data access. It is essentially the middleman facilitating all communication between the database and a DataSet. The DataAdapter is used either to fill a DataTable or DataSet with its Fill method. After the memory-resident data has been 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.

11.Web Page Programming Options

An e-commerce organization can create data-based Web pages by using serverside 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.



12.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.

13.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". Hence we have designed the project to provide the user with easy navigation, retrieval of data and necessary feedback as much as possible.