A web store based on reusable .NET components

By
Aftab Baig – abg09003@student.mdh.se
Iftikhar Ahmad – iad09001@student.mdh.se
August 2011

Supervisor:
Frank Lüders
frank.luders@mdh.se

Examiner:
Kristina Lundqvist
Kristina.lundqvist@mdh.se
Abstract

The thesis project describes the analysis, process and major factors in development of a simple component oriented Web Shop in ASP.NET. It addresses to the concepts used in the application as well as derivation of technical design and development from the concepts acquired by studying existing approaches.

The report describes a brief summary of existing approaches and related technologies. It also lays the foundation of goal oriented approach by providing an overview of component based software engineering. The basic concepts for modularization were barrowed from entities identification, object models and component models. Application’s architecture is set to be a layered approach combining the software layered architecture approach with multi tier architecture of web applications. Class models explaining the inner structure of each component have been provided and an overview of user interface pages is given to explain the application outer flow. The application sets out to prove the significance of component oriented approach as well as the support provided for it by ASP.Net. The resulting package proves to have scalable components that could be scaled for or reused in another application or in a later version of the same application.

**Keywords**: Web Shop, development process, component based, ASP.Net, modularization, Layered Approach, multi-tier approach, scalable, reuse.
A Thanks........

A hearty thanks to Frank for guiding us through our degree, a cheerful gratitude for Asad Iqbal for all the help he gave us with the technical stuff we had to figure out, a happy jovial to Hasan Ali for the help in written communication skills he equipped us with and last but not least, a good bye thanks to the summer in Sweden that almost made us forget the scorching heat that lingers back home every summer.
CONTENTS

Chapter 1  Introduction ................................................................................................................. 6
  1.1  Problem Statement .................................................................................................................. 7
  1.2  Goal ......................................................................................................................................... 8
  1.3  Expected Results .................................................................................................................... 8
  1.4  Web Shop Requirements ........................................................................................................ 8
  1.5  Component Based Web Shop ............................................................................................... 9
  1.6  Scalability of Components .................................................................................................. 10
  1.7  Structure of Report ............................................................................................................... 10

Chapter 2  Background .................................................................................................................... 11
  2.1  Related Work ....................................................................................................................... 13
  2.2  Existing technologies and Approaches ............................................................................... 13

Chapter 3  Component-based Software Engineering ..................................................................... 14
  3.1  A Component ....................................................................................................................... 15
  3.2  Interfaces ............................................................................................................................. 16
  3.3  Reuse of Components .......................................................................................................... 16

Chapter 4  Web Shop Components: Conceptual Derivation ............................................................ 18

Chapter 5  Application Design ...................................................................................................... 23
  5.1  Database Design ................................................................................................................... 26
  5.2  Database Diagram ................................................................................................................. 29
  5.3  Client and Server Sides ........................................................................................................ 30
  5.4  Order manager Component ............................................................................................... 33
  5.5  Payment manager Component .......................................................................................... 35
  5.6  Product Manager Component ............................................................................................ 37
  5.7  Shopping Manager Component .......................................................................................... 39
  5.8  User Manager Component .................................................................................................. 41
  5.9  Extended Classes ................................................................................................................. 43
  5.10 Graphical User Interface .................................................................................................... 49
      5.10.1 User Pages .................................................................................................................. 49
      5.10.2 Administration Pages ............................................................................................... 57

Chapter 6  Implementation with Visual Studio 2010 .................................................................... 64
  6.1  Developing a Web Application ............................................................................................. 65
Chapter 1  Introduction
The business world is changing by rapidly integrating with information technology. Where a few decades earlier, TV and radio were used to promote a business, now internet has become an easily accessible and affordable preference. It’s easy to access, a business owner gets to have customized internet front for his or her business that not only works as an advertisement but also as a whole business in it, where sales and purchase transactions happen almost as frequently as in real, physical dimensional business environment. Sometimes the ease of access linked with internet makes internet a better way for business transactions. Nowadays people can pay their bills from their homes or work places instead of going to any related place to submit the fees. A person can buy a house and fill it in with furniture of choice without even going to a single store. “For customers it is not only because of the high level of convenience, but also because of the broader selection; competitive pricing and greater access to information [1.1] [1.2]. For organizations it increases their customer value and the building of sustainable capabilities, next to the increased profits [1.3]. All this is a part of, what is called as e-commerce uproar.

For all above reasons, every company that sells something has a web site. People can visit the web site, browse around every available product and order any if they come to a decision from the vicinity of their homes. Pizza Hut was the first business to introduce an online shopping system and in their case it was a Pizza shop [1.5].

The basic entities of these Web Shops are customers and products. Then the processing is shifted towards orders and purchases. A simple website may represent several products on one or more pages, an upgraded one will use a database system to maintain synchronized records for these products. A better one would use multi-tier architecture and a flexible design model. But still most of the web shops today are not easily scalable. The generalization is hard and development and maintenance times and cost are higher.

Therefore to make a web shop that not only has all the features of traditional web shops but is also scalable, flexible and reusable, we imply the component oriented development methodologies. The web shop should be able to handle increase in work load; its modules could be reused in future and should have open portals to incorporate changes in future.

1.1 Problem Statement

The purpose of the project that is addressed in this thesis report is to develop a Web Shop in ASP.net using component oriented methodology and by thus coming up with a set of reusable .NET components that can be easily scalable for further use later on.
1.2 Goal

The Goal of this project can be divided into sub goals. First would be to develop a working Web Shop that illustrates a real world business. Our second can be labeled as development of a component based system which could facilitate scalability.

With accordance to the Web Shop development, we intend to make it a good looking, proper web application that satisfies a customer when used and in this regard we will try to fulfill all the requirements for this application as stated in section 1.2. On the account of our second goal we will try to prove the usability, cohesiveness, independent development and reuse of components in ASP.Net by implementing a simple web shop. For these purposes, we will design and develop a web shop. We will reuse any existing components in cases where the required functionality is satisfied by already made components but in other cases we will create required components. The resulting components are aimed to be scalable and reusable for future purposes.

1.3 Expected Results

The resulting project and its report will be able to provide an analysis of component based web application development in ASP.Net along with an example of a simple web shop to demonstrate the said results.

1.4 Web Shop Requirements

As an application, the web shop’s general requirements could be summarized as follows:

- It should be able to handle user activities, their data and access rights.
- It should welcome a user at first access in any session that implies that it should have a welcome page.
- It should cater two user types i.e. customer and administrator.
- For an administrator, it should provide options to:
  - View:
    - Customers.
    - Products.
    - Product categories.
    - Order Histories.
  - Add:
    - Customers.
    - Products.
    - Product Categories.
  - Delete:
    - Customers.
- Products.
- Product Categories.

- For a customer, it should have options for:
  - Registration.
  - Browsing and viewing all products.
  - Browsing and viewing by their categories.
  - Adding products to shopping cart.
  - Editing their shopping carts.
  - Deleting a product or more from their carts.
  - Making an order for products.
  - Making a purchase via online bank transactions.
  - Viewing user’s own history (previous orders).

- It should be component based.
- It should have reusable components.
- It should be able to maintain a well designed database for the company.
- The application should have scalable components for further usage.

1.5 Component Based Web Shop

“A component is an encapsulated piece of code that performs some function for an application [1.5].”

In our application we will separate functional modules based on their internal coupled functional procedures. These modules will be implemented as components so that they can manage their respective functions independently from other components. By this we can make sure that if any kind of change is needed in one of them, it could be implemented, at first with minimal rework and at second, without affecting other components.

This way we can make sure of the increased productivity of the system, its quality and time to market. Maintenance time and costs are also decreased. Overall, it will help us to create an application suitable for strategic business aiming for expansion and extension in near future.
1.6 Scalability of Components

Scalability is the ability of a system, network, or process, to handle growing amounts of work in a graceful manner or its ability to be enlarged to accommodate that growth [1.6].

We are aiming to make the components able to handle any number of users at the same time. Of course this will also depend on hardware involved but from our perspective, the application in general and each and every component in turn should be able to utilize the maximum available resources and by doing so, should be able to handle increasing number of users without any system glitch.

On the other hand, these components should be able to incorporate new functionality or new technology with bare minimum change in existing system. If another kind user is added in the future then only User manager component will need re-work, rest of the system should go unscathed.

1.7 Structure of Report

This report contains 7 chapters.

Chapter 1 gives an introduction to the thesis report giving a brief overview of the factors that lead to the requirements for this project. This is followed by a problem definition, our goals, a statement about expected results and requirements for the application.

Chapter 2 gives brief background information. It flies over the concepts of online shopping, internet trends, web applications and their historic impact on business. It also provides a brief discussion about related and existing projects.

Chapter 3 covers an introductory discussion about components, interfaces, component based software engineering and reusability of components.

Chapter 4 provides Conceptual derivation of Web Shop components.

Chapter 5 covers the Application design, its database concept, its client server side separation, the components of the system with their class models and introduction to graphical user interfaces.


Chapter 7 concludes the work done with emphasis on the scalability of the components.
Chapter 2  Background
Online shopping is a form of electronic commerce. "It is a process where consumers directly buy goods or services from a seller in real-time, without an intermediary service, over the internet [2.1].

For the purpose of online shopping we have online shops which are also known by names like e-shop, e-store, internet shop, web shop, web store and virtual store or virtual shop, where everything that can be done in a same store in real world can be done over the internet. When a customer buys from a web shop, the process is called Business-to-Consumer (B2C) online shopping while when a business buys from another business, it is called Business to Business (B2B) online shopping [2.1].

In 2006, in USA the Business to Consumer product sales reached up to 146.4 billion dollars, representing about 6% of retail product sales in the country [2.1].

The early start of web development points towards static HTML pages where change in the content of a single page could mean changing the whole web site. Dynamic web development was introduced then to eradicate this problem where header files were used that would upload all the files about change in one of them at loading. Common Gateway Interface was established to provide dynamic web pages that could reflect user input [2.3]. Developers and Business holders needed better, more secure, fast to deploy, flexible to change and scalable systems for the increasing need of flow of information over the internet. Therefore as a result, a lot of solutions were provided and are still being implemented and improved. The benchmarks could be pointed out as content management systems, thin client systems, multi-tier architectures and component based development systems. The early methodologies of web development could not manage complex and very high frequency dynamic systems. Nowadays sound scientific, engineering and management principles are implied for web development [2.4]. Different models have been proposed overtime but they usually center around three aspects namely the presentation, the processing and the data.

The separation of the whole system into sub-systems or layers helps compartmentalize each part and serves as protection against impact of change or a fault in another part. The data related module or layer can data and data alone and hence is neither need to change nor is effected when either of the other two parts change, at least not by a long shot. Same goes for the other two layers and even if some change in one has to have an effect on any other layer, its effect is not only minimized but is also traceable, forward and backward both ways. Although, it will highly depend upon the quality of development and the developers ability to trace it but still a well written code can actually guide a developer through its line revealing anything that’s needed to be improvised.
2.1 Related Work

Since we have already established that almost every business has a web shop for their e-commerce purposes, we can start listing them but they will never end. We can start from Pizza Shop, the first online web shop developed for Pizza Hut in 1994. Amazon.com was launched in 1995 and eBay was introduced in 1996 [2.5]. Although most of them are not open source to be examined from a developer’s perspective but still a little of analysis can at least provide an idea about the functional flow of the application. Putting this, some database designs and some research in related technologies a general conceptual prospective can be developed that can prove very helpful in later stages of development.

Some open source web applications are worth mentioning here. OpenCart, Batavi, osCommerce, OXID eShop Community Edition, PrestaShop [2.6][2.8][2.9][2.10][2.11] and phpShop [2.12] are some good examples.

2.2 Existing technologies and Approaches

Most of the open source Web Shops are developed in php because of its internet targeting but they can be made in a lot other languages and tools. We can name following as some major developing tools:

- PhP.
- ASP.
- ASP.NET.
- Pearl.
- C\CGI.
- Ruby.
- Ruby on Rails.
- Java.
- Cold Fusion.

Then we have architecture like MVC model [2.13], Three Tier Architecture [2.10], Component-based Scalable Logical Architecture CSLA [2.11].
Chapter 3  Component-based Software Engineering
Component based software engineering is a branch of software engineering. It emphasizes the separation of concerns with in the give range of functionality provided by a system. Following this approach a system is divided into components based on the cohesiveness of functional units together with the loose coupling that deems a set of functions to be put together [3.1].

A system is divided into separate components based on the functional cohesion and relativity of data. Each component represents a part of the system functionality and keeps the procedures and data related to that part with in itself, encapsulated and isolated from the rest of the system as in classes.

The idea that software should have components was addressed by Douglas McIlroy in his speech “Mass Produced Software Components”, at NATO conference on software engineering in Garmisch, Germany in 1968. And by implementing pipes and filters into unix operation system, he provided the first implementation of this idea. BradCox developed Objectic C programming language to support component based development [3.2]. In early 1990s, IBM developed System Object Model and then actual component software were implemented by Microsoft with OLE and COM [3.1] [3.3].

### 3.1 A Component

A component in literal meanings would stand as part of something or a part which composes something. In software terminology, a component would be a module that encapsulates a set of related functions and/or data [3.4].

According to Bosch, a software component could be defined as:

“A software component is a unit of composition with explicitly specified provided, required and configuration interfaces and quality attributes [3.5].”

An easy understanding would be to relate a component with the pre-existing concepts of objects and object oriented programming. There is a fundamental difference between the two but the derivation of the concept of component could be related to the fundamental divide and conquer technique of software engineering and encapsulation done by objects and classes alongside the concept of hardware components [3.2].

Just like on a circuit chip, a set of IC’s work together, components work together to make the system run and as those IC’s can be replaced by more suitable ones so should the components be able to.

And just like how classes encapsulate data and functions based on real life objects and interaction, components should also encapsulate functionality but instead of basing it on objects, here we base this division and encapsulation on functional cohesiveness.
A component should be considered larger than a class, it should be taken as a package, a whole system that can be deployed independently. It could be a database, an object, a process, a commercial product and may be more [3.3].

Components could be divided into classes according to Szyperski [3.4]:

**White Box:** Total visible and editable implementation.

**Black Box:** Nothing visible except interfaces which are used to communicate with the component. Interfaces are defined in next section.

**Glass Box:** Visible but not modifiable.

**Gray Box:** Parts of implementation is kept visible.

### 3.2 Interfaces

A component uses interfaces to communicate with the outside environment. An interface could be considered as a medium through which it communicates, ask for services or provides services to and from other components.

According to bosch, there are three kinds of interfaces: [3.5]

1. Provided.
2. Required.
3. Configuration.

**Provided** interfaces represent the set of services a component can provide.

**Required** interfaces show the services needed by the component.

**Configuration** components are used for component adjustments into the system.

### 3.3 Reuse of Components

Reusability is considered to be the most important feature of Components. In literal terms reuse would imply using an existing component in a new context, a new system or even within another bigger component. So reusable components are those components which have the ability to be used in other systems, products or later version of same products [3.5].
The benefits of reusable component are directly proportional to its development. The more reusable a component is to be made, the more effort it takes to do so. It will take a lot of awareness, generalization of system requirements, knowledge of possible target platforms, a prediction of change in technology and a very robust development technique to make a reusable component. But if developed, it saves development time for new systems that it is to be a part of, saves development costs for itself and since it is already checked and run previously, so only its compatibility is tested again.
Chapter 4  Web Shop Components: Conceptual Derivation
For the purpose of component based development, the web shop here presented was modularized in to components that can independently work on their own as well as provide ample support for the final system. To do this, the concepts were derived from standard definitions like that of Grady Boch from Software Components with Ada, 1987, stating a component as “ A reusable software component is a logically cohesive, loosely coupled module that denotes a single abstraction [4.1] [4.2].”

Presently almost every company that sells something has a web site for sales. Most of them apply various different technologies depending upon the required functionality and the non function requirements provided by the business stakeholders. Most of them differ from each other in various aspects at the same time have quite a few similarities. But the conceptual layout for most web shops relays almost the same basic decomposition of applications on the conceptual, architectural and design level. A general analysis of web shops like Amazon, e bay. Ellos.se and a few similar others, reveal a pattern that addresses to the major perceivable entities on a general level. Putting in our project requirements, we can deduce some factors that influence the component building of a project and in turn our own project.

First and foremost is the customer or a user, although as a standalone entity from the real world, it could be taken as an object and hence classified as a class but in order to have an independent part of most web shops, a customer manager is implied to handle everything related to customers. On the same line another kind of user exists that is required to access the system’s functionality in a whole different way, which would be an administrator. So a User manager handles, at first the distinction between the two users and then the distinctive functionalities provided to both kind of users as well as the ones that are same for either type.

Then come the products and their management. A product normally comes with general attributes like, name, type, some kind of identification tag and price. A product can be sold, bought, inspected, viewed, and returned in general perception. In a digital system, functionalities like adding, updating and deleting are generally considered the basic functionalities that are expected to be provided. So a product manager deals with all the attributes detailed under products and their categories.

By putting together a customer and one or more products we get an order. An order in general comprises of a purchase by a customer for one or various products. It includes data like purchase date, product ids, their respective quantities, their individual prices as well as the accumulative amount. In web shops, a customer’s identification is usually attached with in an order.

A payment is made for a placed order by a customer. It involves a bank transaction and hence includes back account number related to a user and the related data attached to it like a pin code, bank’s name or some sort of identification for it. Some data representing the underlying order as well as the amount paid plus the transaction dates for record keeping.

Now if we put all of above together, we get the activity named “shopping”. A fellow friend defined shopping for us like"Shopping is: a customer making a payment for an order placed for one or more products, and then getting them of course [4.3].” This activity may include all of the above mentioned activities. These dependencies could not be separated under this module without having an effect on
the system layout. But regardless of the dependencies, here an abstraction of loosely coupled functionalities could be recognized. A component which deals with the merging of related data resulting from separate components as well as maintains the business logic for interconnectivity between these sets of data seems essential here.

Based on above mentioned descriptions we take the first step to establish the conceptual compartments for the system. We have entities like user, product, order payment and shopping so we draw a conceptual deduction as:

Fig. 4.1: Conceptual derivation 1.

Then we take next step as:

Fig. 4.2: Conceptual derivation 2.

And by the way we described the shopping process, we draw:
Fig 4.3: Conceptual derivation 3.

Here by combining these concepts we draw a general diagram to show these entities together:

Fig 4.4: Conceptual derivation 4.

Based on above mentioned conceptual derivatives, we define the basic conceptual components for our system as:

- Product Manager.
- User Manager.
- Order Manager.
- Payment Manager.
- Shopping Manager.
Chapter 5  Application Design
The application’s basic architecture design follows a layered approach by keeping user at the top layer and machine resources at bottom.

![Diagram of application basic design]

According to the requirements of the project, we needed the components to work independently and effectively. So we decided that, in theory, each interface or web page will interact with each component separately and in turn each component can separately access the database as per requirements of that time. For this purpose we concluded on a layered approach based on user-system and system-system interactions, separating the parts of system in layers based on loosely coupled functionalities.

![Diagram of layered approach]

Each page can interact with several components based on the functionality required per user request. In a very same manner, each component has its own module for interacting with data base. By this we achieve a loose coupling in the system as well a cohesive attribute that upkeeps the integrity of each module giving system a security and flexibility.
Since our system like all other systems that interact with a database and whose functionality strongly depends on the data stored, can basically be labeled as an information management system therefore we look at it in a way that provides a standard integrity for such system. One good way is a tired architecture.

By separating the presentation, the processing and the data management into different logical processes, the application becomes more flexible and reusable. Since a web shop usually follows a client server architecture in which we put the user interface code in a separate module, the logic processing modules separately and data storage and management separately. “The concepts of layer and tier are often used interchangeably. However, one fairly common point of view is that there is indeed a difference, and that a layer is a logical structuring mechanism for the elements that make up the software solution, while a tier is a physical structuring mechanism for the system infrastructure [5.1] [5.2].”

Therefore integrating the concepts of component based developing with that of client server architecture and separating it into three tiers, we get a good design for our application.

![Three tier architecture in Layers.](image-url)
5.1 Database Design

First we have a user which is a standalone entity from the real world, it could be taken as an object and hence classified as a class and since it comes with a set of data attributes we can make a data set and in turn a data table for it.

For a User, we need to store some general attributes like a first name, a last name, an address, a phone number and a mobile number. Since we have two kinds of users that are customers and administrators therefore we introduce another attribute user type named “UserType” to distinguish between both of them. We also needed a password for each user so that they can login to their respective account. Then we need a unique identification for each of them so we add an attribute named “ID” that makes the primary key for our user table.

![tbtUser](image)

Fig 5.4 tbtUser (Data table for Users)

Then we have the products. A product normally comes with general attributes like, name, type, some kind of identification tag and price. So we add a table named “tblProduct” to our database design for products. Since many products can be of the same type and there could be various types of products therefore we introduce another table named “Category” to help uniquely identify each product.

For “tblProduct” we use attributes like product name, art number, description and price. To tackle a product’s relation to it’s category we add a Category ID to the table. For each product we have a category Id that points to a category in category table. Lastly to uniquely identify each product, we use a product identification named “ProductID”.

For “Category” table, we use “CategoryID” as the primary key. Other than that we have a category name and its details. The figure below shows both tables with their relation including their respective primary keys as shown in visual studio.net.
Fig 5.5: tblProduct and Category (Data tables for Products with their entity relations)

An order generally comprises of a purchase by a customer for one or various products. It includes data like purchase date, product ids, their respective quantities, their individual prices as well as the accumulative amount. In order to eliminate redundancy we divide the data related to each order in two separate data sets. First one would be “tblOrder” that stores the data like order date, total amount, identification for the bank that is used to pay the mentioned amount, identification for the customer that is “CustomerID” which relates to the “ID” in “tbtUser”. Lastly we use “OrderID” as the unique primary key for “tblOrder”. Secondly we create a table named “OrderDetail” to keep the data related to products of the same type and by such we prevent data redundancy. This table holds attributes like “ProductID”, number of products of same type as “Quantity”, “OrderID” to relate to the order and “OrderDetailID” as a primary key.
Fig 5.6: tblOrder and OrderDetail (Data tables for Orders with their entity relations)

A payment is made for a placed order by a customer. It involves a bank transaction therefore we introduce a table named “Bank”. This table keeps the record of banks related to each user and hence includes a user identification “UserID”, bank account number as “AccountNumber” related to a user and the related data attached to it like a pin code “PinCode”, bank’s name as “BankName”, an attribute “TotalAmount” depicting the amount in the account and an identification “ID” for uniqueness.

Fig 5.7: Bank (Data table for Banks in Payment management component)
5.2 Database Diagram

Figure below shows the entity relation diagram for the database of the system in whole.

Fig 5.8: Entity Relation Diagram for Web Shop Application.
5.3 Client and Server Sides

The development of the project was taken forward according to the concepts provided by the theoretical basis of web application engineering principles as well as those of component based development procedures. In a general web shop, separating User interface pages from rest of the procedures is considered to be a good practice. We name this division as server side and client side. Namely we put it in layers like presentation, business logic and database. In a three tier architecture we put presentation layer in form of web pages that are graphical user interface. These pages reside on server side but are loaded on client side when requested. The business logic and others rules that work behind the scene to achieve the desired results fall in the second layer. The data related to the application is contained in databases on servers.

The business logic part resides on server side too but is hardly ever needed to be loaded anywhere else. Its separation grants a certain amount of security to the system. First we get a system that is robust against a required change. For example in case we need to change the graphical user interface, we need not to change anything other than the web pages and may be a little rewriting of the calling procedures. Then in case if we need to write a new procedure or change an existing one, we only have to cater the situation within the given layer. So goes for the database layer. We access it through the logic layer and by doing so we limit the function of a database as a storage unit and nothing else.

Although we get a good amount of scalability from above features but in order to make it more scalable and flexible, we embed component based technology within these client server and multi-tier patterns. We divide the business logic into several components based on their functional coupling to make each basic function contained separately for further use.

<table>
<thead>
<tr>
<th>Presentation Layer</th>
<th>GUI</th>
<th>Web pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic Layer</td>
<td>Business Rules</td>
<td>Components</td>
</tr>
<tr>
<td>Data Layer</td>
<td>Database</td>
<td>Database</td>
</tr>
</tbody>
</table>

Fig 5.9: A three tier architecture transacting to encapsulate component based development.

As a result we make a component responsible for one functional entity in its entirety. Here, as the names describe, we have a component handling its own transactions, separated from the rest of the system and in this way, a change in one doesn’t affect the other.
Fig 5.10: The design Layout with Components.

The screen shot below simply describes the support for above mentioned methodology in ASP.net and visual studio.
Fig 5.11: A screen shot of the solution explorer describing the separation of components from the GUI part of the application.
5.4 Order manager Component

On the surface the Order Manager deals with the orders made by the customers. It contains the rules related to the formulation of orders as prescribed by the business rules. At one end it helps create and maintain orders for a user and on the other end it covers and communicates with the database sets related to Orders. We can look at it as a component that handles the data and procedures related to all the orders.

Order manager contains following sub-components to help it provide the intended functionality.

"ds_OrderManager" sits in the centre of Order manger. It is a class of type dataset and contains a few other object to perform all the functions that it is supposed to do. It brings in the results of a request and stores them as a dataset to be provided to the requesting part of the application.

We also use IOrderDetailDataTable to cover up for OrderDetailDataTableAdapter class. tblOrderDataTable provides the functional coverage for tblOrderDataTable.

In this and every other component, a whole lot of code is generated by the Visual Studio that includes a lot of predefined classes and libraries and this is the far most fetching feature of new IDE’s.
The figure below describes the class diagram within the order manager component.

Fig 5.13: Order Manager Component Class Model.
5.5 Payment manager Component

The payment manager handles the payments made to make purchases. It keeps the track of user bank accounts and is responsible for all the transaction that is made regarding any purchase. It has following major parts in it’s component infrastructure.

We have two interfaces here, “IDS_PaymentManager” and “ItblOrderDataTable”. The provide the visible connections for other components to communicate with Payment Manager.

Below is the Class Model for Payment Component.
Fig 5.15: Payment Manager Component Class Model.
5.6 Product Manager Component

Product manager handles data related to products. Its main purposes include adding or deleting a new product to and from the database, updating any value related to any product, editing any details about any product and also bringing requested data about any products as per request.

The basic parts of Product Manager are shown below.

Class “ds_ProductManager” is the main class while we use interface “ItblProductTableAdapter” and “ICategoryDataTable” to provide the functionality to other components. Figure 5.17 shows the Class Diagram for Product Manager Component.
Fig 5.17: Product Manager Component Class Model.
5.7 Shopping Manager Component

The shopping manager handles the interconnection of orders and clients (users). It uses order history and user history to formulate the data for shopping criteria. Its basic components are as follows and figure 5.19 shows the class model for shopping manager.

![Diagram of Shopping Manager Components]

Fig 5.18: Parts of Shopping Manager Component.
Fig 5.19: Shopping Manager Component Class Model.
5.8 User Manager Component

The User Manager handles the procedures related to Users. First it distinguishes the type of user logged in which basically constitutes the access level of a user in the overall system. Then it also contain the user data, user history, orders, shopping, contact information and similar stuff. It uses a class “dsUserManager” of type DataSet, an interface to provide communication facilities with any requesting part of the system and event handlers to cater any change in stored data related to a user.

Its basic parts are as falls and figure 5.21 shows the Class Model for User Manager.

![Diagram of User Manager Component]

Fig 5.20: Parts of User Manager Component.
Fig 5.21: User Manager Component Class Model.
5.9 Extended Classes

We have already discussed the design derivation for this project in previous sections including class diagram models for each individual component separately. In the next section we will combine these components to draw the overall implementation design diagram although we will keep the internal structure of components abstract for the sake of drawing the model on a single visible page. After that we will put snapshots representing classes and other parts of each component as drawn with in visual studio.
Fig 5.22: Order Manager Component Parts.
Fig 5.23: Payment Manager Component Parts.
Fig 5.24: Product Manager Component Parts.
Fig 5.25: Shopping Manager Component Parts.
Fig 5.26: User Manager Component Parts.
5.10 Graphical User Interface

This section describes and shows the GUI and its several pages that are used to interact with users.

5.10.1 User Pages

These are the web pages that a “Customer” type user can access and use to shop around.

5.10.1.1 Welcome Page

This is the first page that appears when a client or user accesses the main web site. It is same for both customers and administrators.

![Welcome Page](image)

Fig 5.27: Welcome Page.

The page below is basically a Login page and is similar to both users and administrators.
Fig 5.28: Welcome Page for users to login with their respective accounts.
5.10.1.2 Registration Page

The Registration page is used for new users to get registered with the shop. Their data is provided here and then is inserted into database.

![Registration Page](image)

Fig 5.29: Registration Page.

5.10.1.3 User Account Page

User account page lists the account catalogue for each ser. It shows the orders by dates including the items parched with their prices.
### User Account Page

**Fig 5.30: User Account page.**

<table>
<thead>
<tr>
<th>Order Date</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>2011-08-15 14:41:18</td>
<td>100</td>
</tr>
<tr>
<td>2011-08-07 16:20:35</td>
<td>100</td>
</tr>
<tr>
<td>2011-05-08 15:56:52</td>
<td>550</td>
</tr>
<tr>
<td>2011-05-08 15:55:36</td>
<td>100</td>
</tr>
<tr>
<td>2011-05-08 14:12:50</td>
<td>216</td>
</tr>
<tr>
<td>2011-05-07 16:15:23</td>
<td>400</td>
</tr>
<tr>
<td>2011-05-07 16:10:43</td>
<td>100</td>
</tr>
<tr>
<td>2011-05-07 15:07:18</td>
<td>300</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Order Date</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>2011-05-08 15:56:52</td>
<td>550</td>
</tr>
<tr>
<td>2011-05-08 15:55:36</td>
<td>100</td>
</tr>
<tr>
<td>2011-05-08 14:12:50</td>
<td>216</td>
</tr>
<tr>
<td>2011-05-07 16:15:23</td>
<td>400</td>
</tr>
<tr>
<td>2011-05-07 16:10:43</td>
<td>100</td>
</tr>
<tr>
<td>2011-05-07 15:07:18</td>
<td>300</td>
</tr>
</tbody>
</table>
5.10.1.4  Shopping Cart

As the name describes, the pad shows a cart where user can add items of his or her liking.

![Shopping Cart page](image)

Fig 5.31: Shopping Cart page.
5.10.1.5 **Purchase Product Page**

The page provides an interface for users where they can purchase a product and pay for them. Figures 5.33 and 5.34 show this process.

![Shopping Cart Page 2](image)
Fig 5.33: Purchase Products Page.
5.10.1.6 Payment Page

Fig 5.34: Payment Page.
5.10.2 Administration Pages

These are the pages only accessible to users with admin rights. Here they can manipulate the data of the system in an easy to go way.

![Image of administration page](image.jpg)

Fig 5.35: Main Section management page 1.

These Pages provide administrators with options to add, edit and delete categories and their details for products. An admin type user can simply click on any edit or delete link label against each present category to perform said operation on it or the user can use “New Category” link label on right-top of the category list to enter a new category.
In these pages, like in the pages mentioned in the previous section, an admin can add, edit or delete product. The page below shows the list of present products in a grid view along with their respective data. Here too, adding a new product requires clicking “New Product” on top-right, just above the product list. A new window appears with proper input tools where admin can input the data related to the product and it is inserted in the database. Figure 5.39 shows the mentioned window.

Figure 5.38 shows the editing part for product where user can simply click “edit” against a product and the row becomes editable with a dropdown menu for category showing the available categories and text boxes for the rest of the attributes.
### Fig 5.37: Product Management Page 1.

<table>
<thead>
<tr>
<th>Category</th>
<th>Name</th>
<th>ProductActive</th>
<th>Description</th>
<th>Price</th>
<th>Image</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The Shining (1980)</td>
<td>1</td>
<td>ranks number 1 on The Moving Arts Film Journal's list of Top 100 Horror Movies of All Time</td>
<td></td>
<td></td>
<td>Edit/Delete</td>
</tr>
<tr>
<td></td>
<td>Airplane (1980)</td>
<td>1</td>
<td>number 1 on statistic William M. Briggs's list of Top 100 Space Movies of All Time</td>
<td></td>
<td></td>
<td>Edit/Delete</td>
</tr>
<tr>
<td></td>
<td>Spirited Away (2002)</td>
<td>1</td>
<td>highest-rated anime film on Anime News Network's Top 500 Anime</td>
<td></td>
<td></td>
<td>Edit/Delete</td>
</tr>
</tbody>
</table>

### Fig 5.38: Product Management Page 2.

<table>
<thead>
<tr>
<th>Category</th>
<th>Name</th>
<th>ProductActive</th>
<th>Description</th>
<th>Price</th>
<th>Image</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The Shining (1980)</td>
<td>1</td>
<td>ranks number 1 on The Moving Arts Film Journal's list of Top 100 Horror Movies of All Time</td>
<td></td>
<td></td>
<td>Edit/Delete</td>
</tr>
<tr>
<td></td>
<td>Airplane (1980)</td>
<td>1</td>
<td>number 1 on statistic William M. Briggs's list of Top 100 Space Movies of All Time</td>
<td></td>
<td></td>
<td>Edit/Delete</td>
</tr>
<tr>
<td></td>
<td>Spirited Away (2002)</td>
<td>1</td>
<td>highest-rated anime film on Anime News Network's Top 500 Anime</td>
<td></td>
<td></td>
<td>Edit/Delete</td>
</tr>
</tbody>
</table>
Fig 5.39: Product Management Page 3.
5.10.2.2 User Management Pages

These pages provide the admin a list of users (clients in this case) to see as well as to edit their details when needed. Figure 5.40 shows the list of customers while 5.41 shows an ‘Edit’ process.

Fig 5.40: User Management Page 1.
**Fig 5.41: User Management Page 2.**
5.10.2.3  Order History Page

The page shows a list of orders with their details.

![Order History Page](image)

Fig 5.42: Order History Page.
Chapter 6
Implementation with
Visual Studio 2010
Microsoft Visual Studio is an integrated development Environment (IDE) provided by Microsoft. It supports development of various types of projects including desktop applications, services, web development and web services. It also supports development for all Microsoft based platforms like windows, windows mobile and .Net framework [6.1].

It includes a strong code editor that supports “intelligence” and on top of that it supports various programming languages like C#, Visual Basic, Visual C++, J#, F# as of 2010 and ASP.Net along with HTML, XHTML and CSS. The debugger included in the editor is a major plus in favour of Visual studio as it makes it possible for a programmer to go through step to step of programming watching each transition and state of data in the program. The Form Designer and Web designer along with class designer and database schema designer makes it very easy for a develop a high performance and robust application in a lot less time than it could have taken if not for visual studio [6.1].

For the sake of a visual studio newbie, an installation guide can be found at [6.5]. Msdn site and others applies visual step by step tutorials for beginners that can prove very helpful for beginners as well as experienced one [6.2] [6.3] [6.4]. Below we will describe very briefly the process of starting a web application project in visual studio 2010. Later on we will brief about adding required parts in it to make the whole web shop. We will keep it at introductory level so that a reader can get an idea about the procession of stages throughout the development process.

### 6.1 Developing a Web Application

A simple step by step walkthrough will be presented here to describe the development stages of a simple web application. ‘->’ would imply that after the arrow is the next step to previous one that would be the one before arrow.

1. **Making A web Project**
   - a. In visual studio click File -> New ->Project.
   - b. In New Project Dialog box, first choose the language of your choice, in our case it is C#.
   - c. Then click Web -> ASP.Net Empty Web Project.
   - d. Choose a name, a directory and the location where you want to save it.

2. **Making a Web Page**
   - a. Now the project has been created.
   - b. Now select the project in solution explorer and click Project -> Add New Item -> Web Form.
   - c. Give a name to your page, in our case we keep it default.aspx.
d. Here we can a few more pages in the same way by repeating b and c but giving them different names, here we name them, UserPage, ProductPage, and OrderPage just for the sake of creating an example.

3. Adding Controls to a Page

   a. Now select default.aspx from solution explorer and drag any control onto the page.

   b. You can see the control on the page. Let’s say here, it’s a label and three buttons. You can select these controls one by one and drag them over the page to relocate them.

   c. When you select a control, in property windows all the property data of the control is loaded and can be change according to the requirements.

   d. Let’s say we want to rename the label here, we can change so by clicking in front of name property and writing the new name say Home here. If we want to change the text appearing on a button from button1 to Product Page, then we remove the button1 from in front of Text property and write Product Page.

   e. Here, we change the text of label as main page, button1 as Product Page, button 2 as User Page, button 3 as Order Page.

4. Making Pages work together

   a. We can put labels on each of the other pages and change their texts according to their names. For example label on Product page will read Product page and so on.

   b. Now select default.aspx from solution explorer.

   c. Click designer view and you can see the controls.

   d. Now select button ”Product Page“ and double click on it.

   e. The visual studio will take you to the code page and where it creates an event handler for the button’s click event.

   f. Here we can add any functionality that we would want to happen when the said button is clicked. Here we want it to take us to the Product Page therefore we add a simple code snippet in the event handler.

     Response.Redirect ( “ProductPage.aspx” );

   g. We follow the same procedure from d and f for the next two buttons and make them redirect us to their respective pages namely UserPage and OrderPage.

Now by setting the default.aspx as home page, we can run the project to see its working. The default page is loaded in the default internet explorer. You will be able to see a label saying Home and buttons
Product Page, User Page and Order Page. By clicking you will be redirected to Product page and same goes for the rest of two buttons.

Sections [6.6] [6.7] refer to walkthroughs provided by MSDN for development of a web application. They were great sources to get us started with visual studio and the above description of the development process is partly based on the descriptions there.

### 6.2 Adding Database

Database is created by SQL server, here we used SQL server 2008 which gives a pretty good Graphical interface to create tables and create relations among them. You can add a new or existing database from Project -> Add new Item -> Data -> SQL Server Database or an existing database by clicking Project -> Add Existing Item and then by browsing and selecting your database. Visual studio provides ample components to help you connect to, use and manipulate your database effectively.

### 6.3 Adding Components

Adding a component is fairly easy in visual studio. By creating a new project like before but instead of making a web project, we select class library from the list and rename it and save it as we please. Here we can add code to this component, add new classes or anything that we need. Then the component can be added the project by selecting the web project, right clicking on it and selecting add and then selecting the required component [6.8].
Chapter 7   Scalability
The scalability was our main concern along with reusability. The system can not only handle as many user accessing it based on the server processing power but can also use as much processing power is provided at any given instant. If the server is provided with new processor, the application will use it. It can work properly with most known internet browsers and with all Microsoft based servers. Also, in case if the company using the web shop adds another shop to its business, we may either only need to add a small new component that would handle the distinction of companies and rest of the system remains the same or we may add an extra attribute to product data base to identify which shop a product is coming from.

When it comes down to scalability of a web site, we think about internet traffic, bottlenecks, available resources, new machinery, added nodes, existing nodes and effect of above factors on the working of the software. The first step that we took was to modularize the whole system. Using architectural approach like inducing layers that separated modules of software and at the same time encapsulated chunks of software that could be loosely coupled together based on their functionalities.

Another step was to utilize the distinction between the user types to our benefit. Although, for now since the anticipated load on system is very low, we haven’t separated the data related to different user types in to different chunks but later on when we need it, a small change in central configuration file will be able to accomplish it in no time. We also used the concepts of cache in order to avoid a failure in any transaction due to any load. Database cache stays invisible to the application but helps maintain the application performance.

The database reader and writers are coded in separate functions, database adapters and data table adapters are used to keep the logical processing separated from database and in case of a new db server or shifting of database on a more powerful machine will only improve the reaction time of database related queries.

Any number of new pages can be added to the application without having to make any drastic changes in the processing or database layers. And any number of users can access the application from anywhere at any time. Defining bottlenecks is usually very difficult but with the overall approach that we used here, the response time of queries can be detected just by looking at the request coming from the page or user interface. Since each component handles its own functionality, we can add resources separately to each component with little or no change in the system.
Chapter 8  Conclusion
The component based approach refines the development process by partially intensifying the functional separation of the modules and partly by providing the higher opportunity for scalability for the future purposes for the project.

ASP.Net with visual studio provides very user friendly environment for development purposes. A graphical user interface helps reduce the coding time and complexity and the built in libraries and components help increase the authentication level of the solution.

The sample Web Shop demonstrates the essential parts of a good efficient Web Application. The component based approach used insures a high level of scalability as well as reusability of various components. The components developed for the project can be used in various other projects which contain the conceptual entities same as this project, these entities would be the user, product, order and payments. These components can be used in most of the projects that deal with these entities.

The scalability was our main concern along with reusability. The system can not only handle as many user accessing it based on the server processing power but can also use as much processing power is provided. If the server is provided with new processor, the application will use it. It can work properly with most known internet browsers and with all Microsoft based servers. Also, in case if the company using the web shop adds another shop to its business, we may either only need to add a small new component that would handle the distinction of companies and rest of the system remains the same or we may add an extra attribute to product data base to identify which shop a product is coming from.

All in all we can conclude that the project successfully fulfils the requirements it was supposed to meet and describes the benefits of component based approach and the feasibility of choosing ASP.Net as the development environment for its assistance rich features.
REFERENCES


