CORE
Metadata, citation and similar papers at core.ac.uk
Provided by K-State Research Exchange

MYBOOKSTORE-ESHOPPING FOR BOOKS

by

SUSHMA REDDY CHITTURI

B.E., Osmania University, 2006

A REPORT

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Computing and Information Sciences
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2010

Approved by:

Major Professor
Dr. Daniel Andresen

Abstract

The Web is a shopper's paradise boasting every kind of product imaginable — plus many more that are almost unimaginable. People find it easy and secure to shop online these days thereby saving time and also have more options to choose from at their fingertips. Based on this comes MyBookStore, a neat web application designed to exclusively cater the needs of students for purchasing books online. Primary focus of this application is to ease the use of searching for a particular book by the user and also navigability within the website. A sophisticated search engine has been designed in this application which filters the products based on various user criterions. Searching and viewing the details about a book is available. This also has an administrator side through which the administrator can update the website with new products, remove any of the available products, and add new categories, subcategories and products along with updating the shipping status of orders placed. This section is majorly responsible for user accounts maintenance, product maintenance as well as orders maintenance. Major emphasis of this application is to build user interactive search techniques for simplifying user needs and to provide specific products as required by the user.

Table of Contents

Acknowledgements

I would like to thank my major professor Dr. Daniel Andresen for his constant guidance and help throughout the project. I would also like to thank Dr. Gurdip Singh and Dr. Mitchell Neilsen for graciously accepting to be on my committee.

Dedication

Dedicated to My Parents and My Friends Sujith and Rohit who have been extensively supportive throughout my Masters.

1. INTRODUCTION

1.1. Goal

The Web is a shopper's paradise boasting every kind of product imaginable — plus many more that are almost unimaginable. People find it easy and secure to shop online these days thereby saving time and also have more options to choose from at their fingertips. Goal of this application is to design and develop a rich user interactive web application for customers to meet their academic book needs. Primary concern would be to make searching of products easy for the users and also to simplify the process of check out. Major goals of this application are:

An **AJAX** enabled web application which has been designed with **AJAX** 3.0 controls and extenders to give a clean as well as rich look to the application by preventing all those unnecessary flickering caused due to post backs.

Efficient search engine that can filter the products based on various user criterions and that can provide accurate results that can serve well as per the user needs.

User navigability through-out the application so that they can reach a destination page without much traversal within the website.

Ability to view complete specifications of the book and the ability for the user to search for a specific item in the cart.

1.2. Motivation

Though there are various online shopping websites not all have been successful in delivering to the needs of the users. Some of the problems that these websites have are:

Use of non-interactive and traditional user interfaces which does not provide rich look to the application.

No effective search methodologies have been implemented through which a user can locate a particular product from various number of products that are available in the website.

Filtering of a search result based on various other criterions so that a user can land at a specific product that he/she has been looking for.

Delay in displaying search results because of post backs to the server which can cause inconvenience to the users and use of traditional interface can cause the web pages to flicker due to the post backs.

No search methods have been implemented to search for an item in the shopping cart.

Solution: MyBookStore is an attempt to address the above stated issues.

Primary goals are:

Use of **AJAX** controls to provide rich and interactive user interface which handles the issues of flickering of the screen because of post backs. Use of view state and sessions to avoid unnecessary calls to the database which can speed up the search results there-by providing faster response to the user. Easy navigation within the website there by avoiding unnecessary traversal in the website. A sophisticated search and filter techniques which enables users to look for a specific product at a faster rate.

The ability for the user to search for items in the shopping cart and the ability to create user accounts and the ability for the users to manage their account details along with the ability to look for the details of the orders placed.

1.3 Scope

Present system can be extended to enable drag and drop of the items into the cart and allowing users to save the items into the cart and can check-out later.

Users can save their shipping and payment information along with the ability to create multiple billing accounts, and also to be able to subscribe for alerts if a new product has been added to the website or if a price drop occurs on a product. For now this system has a simple checkout process where verification of payments has not been handled. This can be extended to handle the verification of credit card payment, to allow registered users to have coupon points which they can use during checkout process to lower the shopping total.

1.4 Platform Specifications

1.4.1 Hardware Specifications

A processor of P IV and above with a minimum 250 MB RAM and space requirements of about 100 MB are required.

1.4.2 Software Specifications

An Operating System of Windows 2000 and above with .Net Framework of 2.0 or above installed with MS Sql Server 2005 and above along with Ajax Framework are minimum requirements for this application to be developed. An IIS server would be required to host or deploy the application. Visual Studio 2005 or above is required for developing/coding the application.

.

2. SYSTEM REQUIREMENT ANALYSIS

2.1. *Information Gathering*

As stated earlier in the motivation for this project a lot of information was collected from various online shopping websites that were available in order to learn the advantages and disadvantage of those web sites and to come up with the features that were developed in this application to handle some of the issues that came up during the research of such online web sites. Along with this research constant discussions with Dr. Daniel Andresen has proven invaluable in implementing the search and filter techniques in this application which is the major goal of this application.

Other than the functional research that was carried out a lot of research has been done on learning and implementing this application using **AJAX** framework controls, web services using yahoo API to locate city and state based on zip code given by a user while filling out shipping information and also on design patterns for developing the database for this application.

2.2. *System Feasibility*

2.2.1. Economic Feasibility

This application is economically very feasible as the only requirements to use this application would be a minimum modern day computer with an internet connection at a good speed (with a modern or with wireless connectivity) web browser such as IE6 or higher, Mozilla Firefox or Safari. To deploy this application

all is required is a computer with above mentioned hardware and software specifications.

2.2.2. Technical Feasibility

Technical details for deploying this application would be

- Operating System of Windows 2000 and later
- IIS Server
- MS Sql Server 2005 or later
- .NET framework 2.0 or later.

Technical details for using this application would be a computer with an internet browser along with internet connection.

2.2.3. Behavioral Feasibility

Most of the features that are developed in this application are self-explanatory hence no special technical expertise is required in order to use this application. Users are directed through the usage of tool-tips, user friendly messages.

3. SYSTEM ANALYSIS

Based on the above research that was carried out in analyzing this application, various entities have been identified and the relationships between these entities have been demonstrated using an

E-R Diagram, Use-Case diagrams, state transition diagrams and data flow diagrams.

3.1. *Use-Case Diagram*

A use case diagram depicts the application from an external observer perspective. In this diagram we identify the agents of the system and the functions that these agents perform with the system. In this application we have two kinds of agents or actors, a user who is interested in online shopping and an administrator who is responsible for maintaining the website. Below are the two use-case diagrams depicting the functions that are carried out by both the actors.

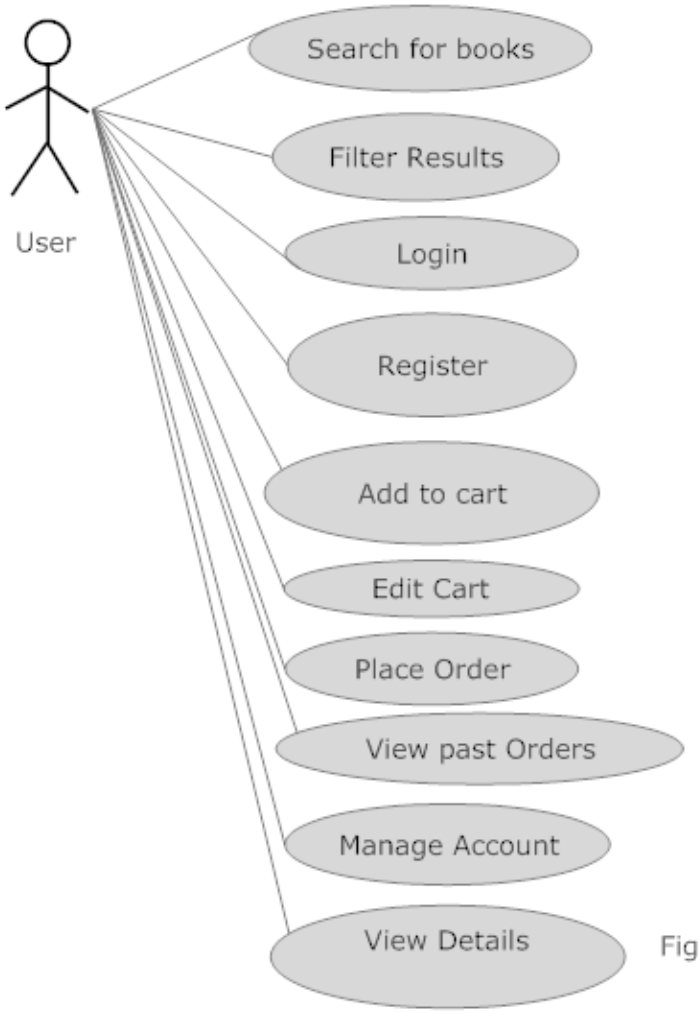


Figure 1: Use Case Diagram for User

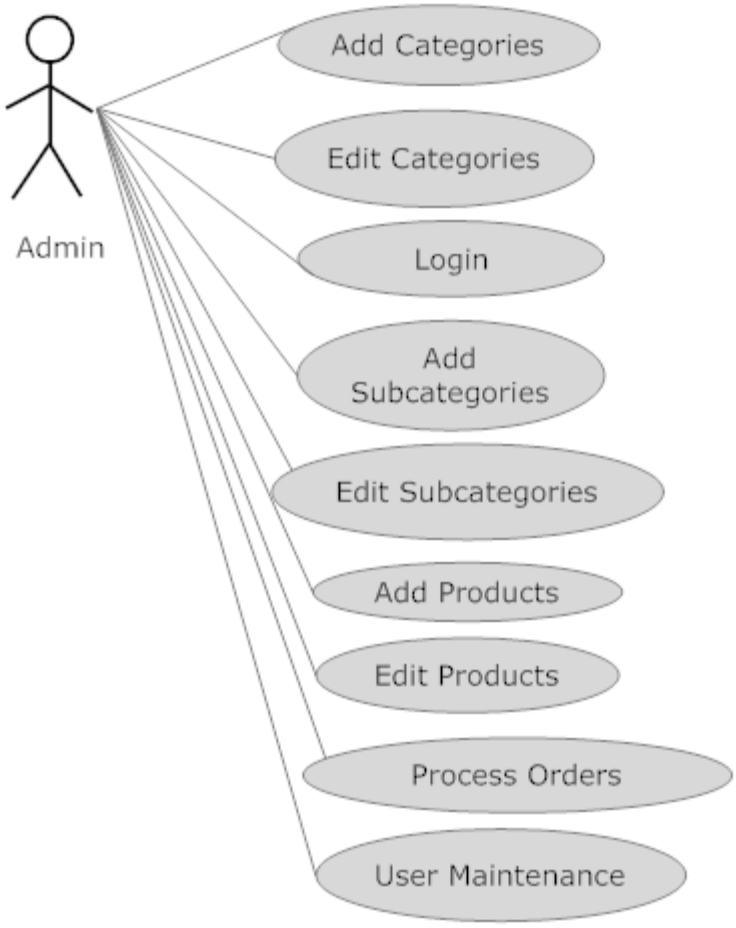


Figure 2: Use Case Diagram for Administrator

3.2. E-R Diagram

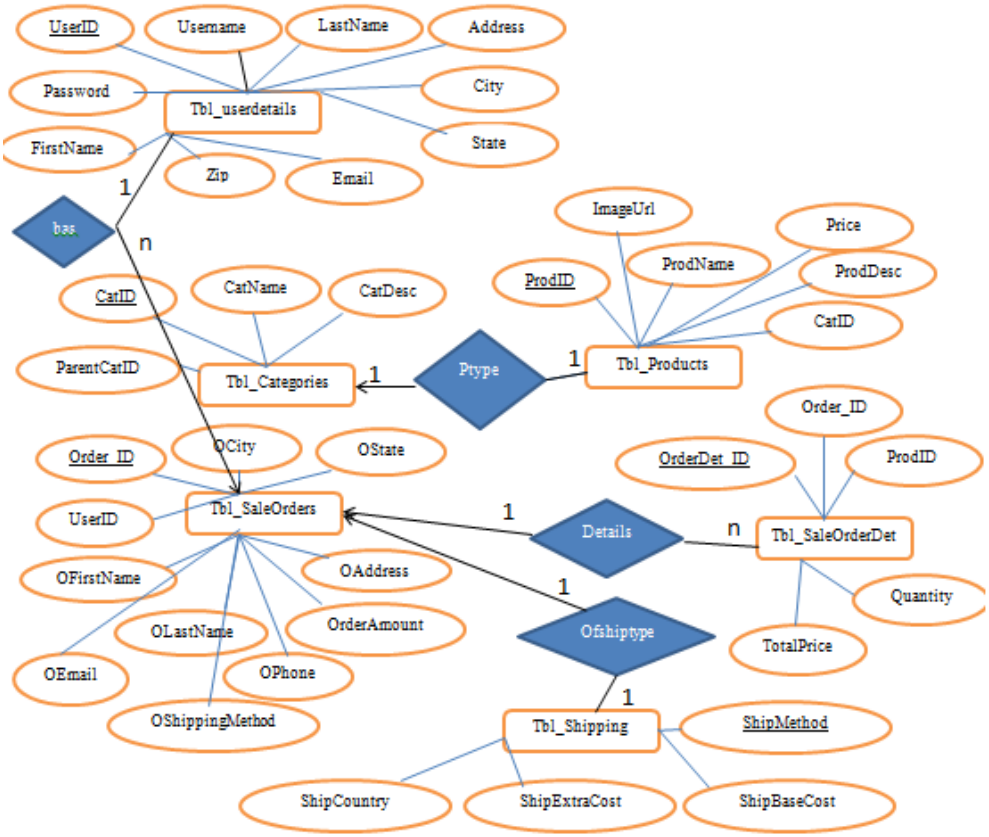


Figure 3: Entity- Relationship Diagram

Understanding the Diagram

The major entities of this application are Tbl_userdetails, Tbl_Categories, Tbl_Products, Tbl_Shipping, Tbl_SaleOrders and Tbl_SaleOrderDet.

Tbl_userdetails: This table holds the information of all the users who are registered with the website using the registration page in the application. All user information such as registered username, password along with Address information is stored in this table and every registered user is given a unique id called as UserID which will be used throughout the application to recognize the logged in user and to track his activity in the application.

Tbl_Categories: This is used to hold the information about the type of categories and sub categories that are present in the application. Each category is uniquely identified by its CatID. ParentCatID attribute determines if that particular category is a parent category or a

sub category. A category which has its ParentCatID value as null represents a main category and a category which has a non-null value

Tbl_Products: This table is used to store all the products. Each product has its unique identifier with other attributes such as its price, quantity and status of the product which means if the product is available or not in the site. Each product is related to a subcategory through CatID attribute in the table which is the foreign key to this table. Tbl_Products and Tbl_Categories are related via **Ptype** relationship. A product can only belong to a single category.

Tbl_Shipping: This table holds different types of shipping options that are available to the user. Each method is uniquely identified and has attributes such as country where they can ship to, it's pricing details.

Tbl_SaleOrderDet: This table holds itemized details about each order. Each is uniquely identified with its own id which has details about each product a particular order has. It has two foreign keys Order_ID and ProdID from Tbl_SaleOrders and Tbl_Products respectively.

Tbl_SaleOrders: This table holds information about all orders placed through the system. Each order is uniquely identified with Order_ID. Orders are related to users through **has** relationship and a single user can have many orders but each order belongs to a single user. Orders and Shipping are related via **ofshiptype** relationship and a single order can have only one shipping type and Orders and Tbl_SaleOrderDet are related via **Details** and a single order can have many entries in the Tbl_SaleOrderDet table whereas each entry in Tbl_SaleOrderDet is related to only one order in Tbl_SaleOrders table.

Entities and their attributes along with relationships among them were carefully deigned such that there is no scope for any redundancy in the data. Tables were normalized such that no duplicate data exists in the database thus making database retrievals, inserts and updates much simpler. Having normalized data has enabled me to write effective stored procedures with less code. For example deleting an order should also delete all the order details in the Tbl_SaleOrderDet and since I maintained foreign key relationship between both the tables deleting an order entry in Tbl_SaleOrders automatically deletes its corresponding entries in the other table with setting a simple property of cascading delete to true. This reduces the effort in writing code to handle cascading deletes and also to make sure that we do not want any unused data in the database thereby having to waste a lot of storage.

3.3. Data Flow Diagrams

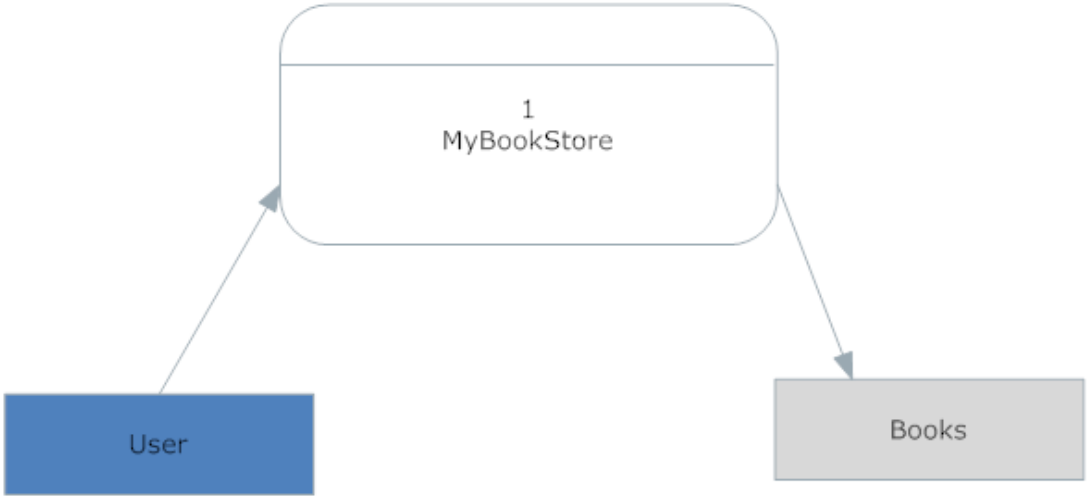


Figure 4 : A Context Level Flow Diagram

3.3.1. A Data Flow Diagram Depicting a User Placing an Order

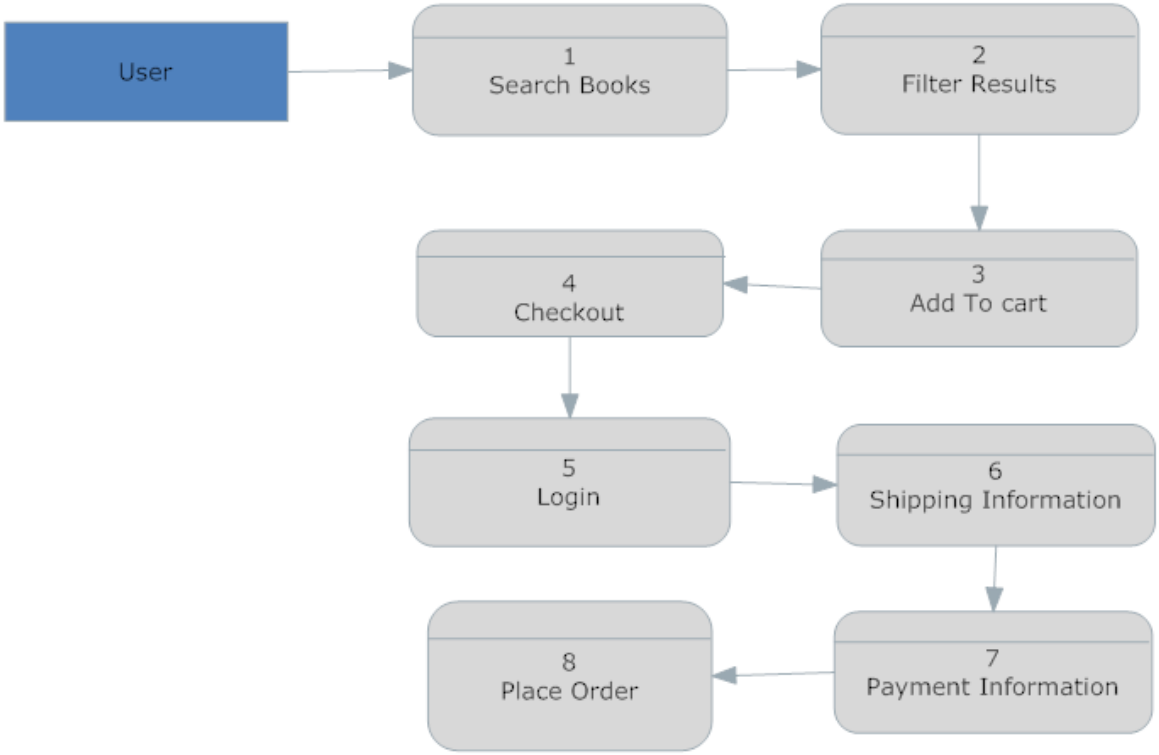


Figure 5: Data Flow diagram for a User placing an Order

3.3.2. A Data Flow Diagram Depicting a User Viewing Details of a Book

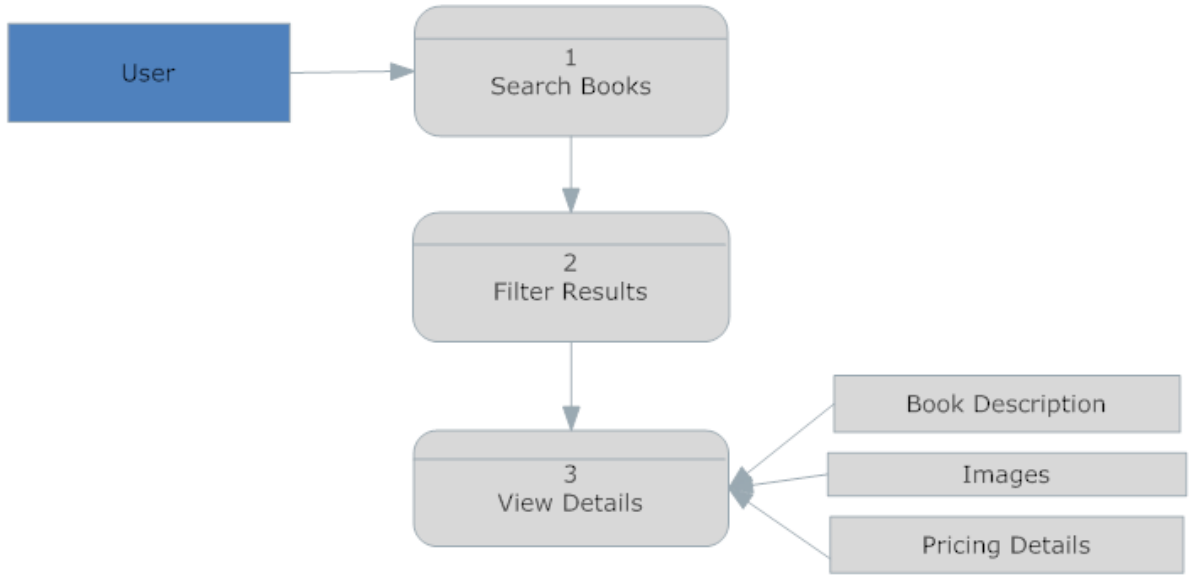


Figure 6: Data Flow diagram for a User Viewing Details of a Book

3.3.3. *Data Flow Diagram Depicting an Administrator Adding a Book to the Store.*

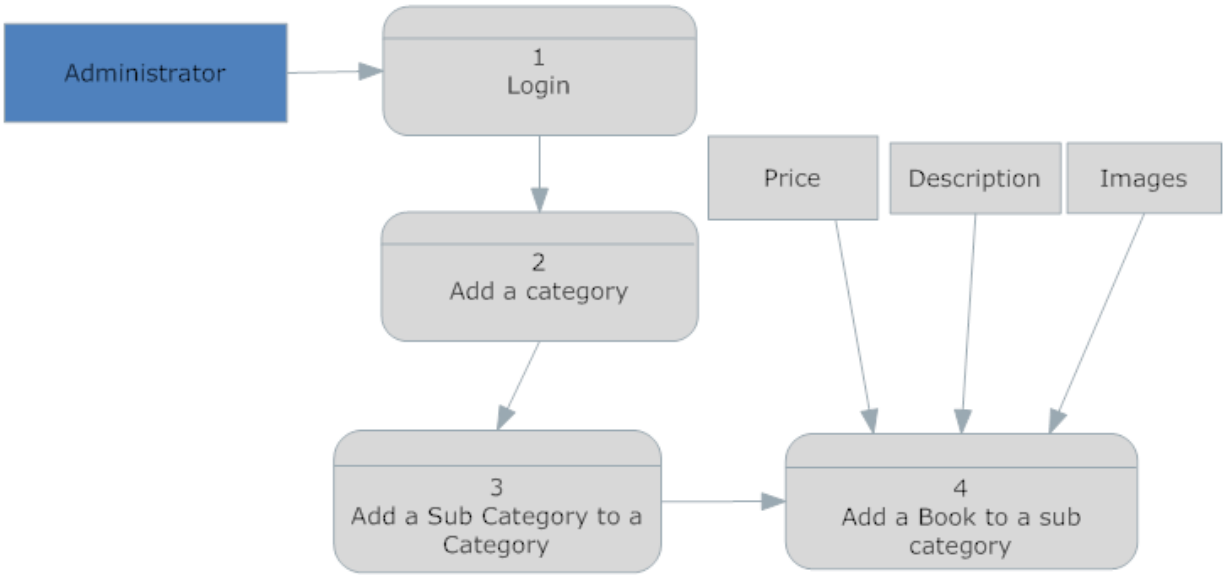


Figure 7: Data flow diagram for an Administrator adding a book to the store

Understanding the diagrams: Figure 5 represents sequence of actions that a user may perform while placing an order with the system. Initial state of the system would be a state where all books are displayed on user's home page. A user can perform a search for books and then he can filter the search results based on various options that are available. After filtering to

place an order he has to add to cart that particular book after which he can check out the cart. But before the checkout process begins a check is made to make sure that user is logged into the system hence he is required to login before he can provide the shipping details and payment information. Once he is logged in and finished entering all required details he can place an order. In order to place an order all the above steps have to be performed from Login state of the system. Another approach can be making the user log into the system at the first place and then only he can access the site but that state would make the application less user friendly because a user would want to register with the site only if he has needs met by the application.

Figure 6 represents sequence of events that can occur if the user wants to view details about a book. In the figure it is shown that he can search and filter the results and then can view the specifications which include description of the book, book cover and pricing details. Another sequence of events might be when user directly can search for a book if he knows the title which directly takes him to the details screen.

4. DESIGN

4.1. *Design Goals*

- Major design goals of this application are to build user interactive web pages for viewing available books, searching for a particular book, viewing the details of the book.
- Design of sophisticated search techniques for filtering the results effectively and fast.
- Design of flicker free web pages during post-backs through usage of **AJAX** framework controls.
- It also involves designing fast responsive web pages through the usage of session and view states thereby considerably reducing the post backs to the server and also calls to the database itself.
- Other design goal is to implement easy navigability throughout the application so that the user can navigate to any page from any page within website.

4.2. *Contextual Architecture*

4.2.1. *Architectural Context Diagram*

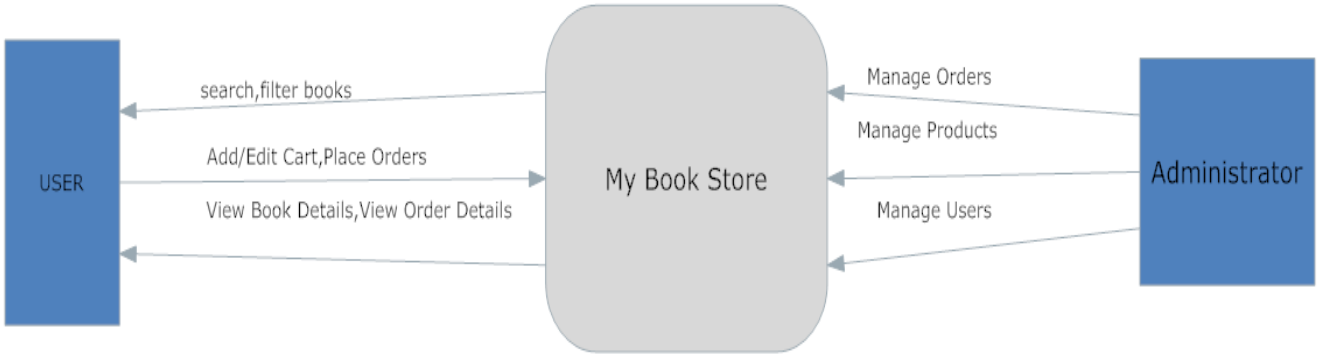


Figure 8: Overview of the Architecture of the application.

4.2.2. *Description of Contextual Architecture*

In the above architectural diagram it can be observed that data flow is in a two-way direction where data is received from the application as well as sent to the application. Boxes represent the points of origin of information and arrows represent the data flow.

User interaction as well as administrator interaction with the application takes place through a graphical interface. Inputs to the application can be a search query term, filter criterion; user credentials and the outputs from the application are displayed on the graphical user interface in the form of grids, dropdowns, data lists which displays the available products, order details, description about the book based on the results that are generated by taking into consideration user inputs.

Modular Approach

This application has been implemented on a modular approach where the application logic has been divided into separate modules so that change to one module doesn't affect the functionality of the others which means all the modules are loosely coupled.

4.2.3. Available Products Module

This is the very initial module that loads when the user visits the home page of the application. All the products/books that are available here are displayed along with the ability for the user to filter the products that come up on his/her homepage. User is also allowed to add products to the shopping cart and here we generate session for the products and this session is maintained throughout. Along with the ability to add the product user is also allowed to view complete details about a product by clicking on the image of the product where the summary of the book gets displayed.

4.2.4. *Cart Module*

This module begins as soon as the user adds a product to the cart and this module is wholly responsible for performing all the functionality of the cart such as updating the cart, handling the total cost of the cart items and viewing the cart items. User can directly update the quantity of the items in the same module and also can remove items from the cart from a single location. From this module the user can check-out the items from the cart.

4.2.5. *Login Module*

This module begins as soon as the user tries to log into the system before he can check-out the items in the cart or when he tries to view his account details or order details by logging into the system. As soon as this module is activated a session will be generated for each user and that session is maintained until the user is logged out of the system.

4.2.6. *Checkout Module*

This module begins as soon as the user session begins after shopping cart module has been executed. This module is responsible for handling the shipping details, payment details; order total based on the shipping method chosen and order confirmation. All this is maintained using sessions for each user. This would be the final module that will be executed by the user for placing an order.

4.2.7. *Administrator Module*

This module gets loaded whenever the admin of the web site logs in to the system and it displays various options to the administrator through which he can maintain the website.

4.2.8. *Categories Module*

This module is responsible for either adding a new category to the system or updating existing categories and gets loaded after the administrator module has been executed by the admin.

4.2.9. *Sub Categories Module*

This module is responsible for either adding a new Sub category to the system or updating existing sub categories.

4.2.10. *Products Module*

This module is responsible for either adding new Products to the system or updating existing products.

4.2.11. *Orders Module*

This module is responsible for maintain orders placed by the customer such as updating the shipping status of the order or cancelling an order.

4.3. *System Architecture*

This application has been designed on the famous three tier architecture. GUI Presentation Layer, Business Logic Layer (BLL) and Data Access layer are the three tiers of this architecture

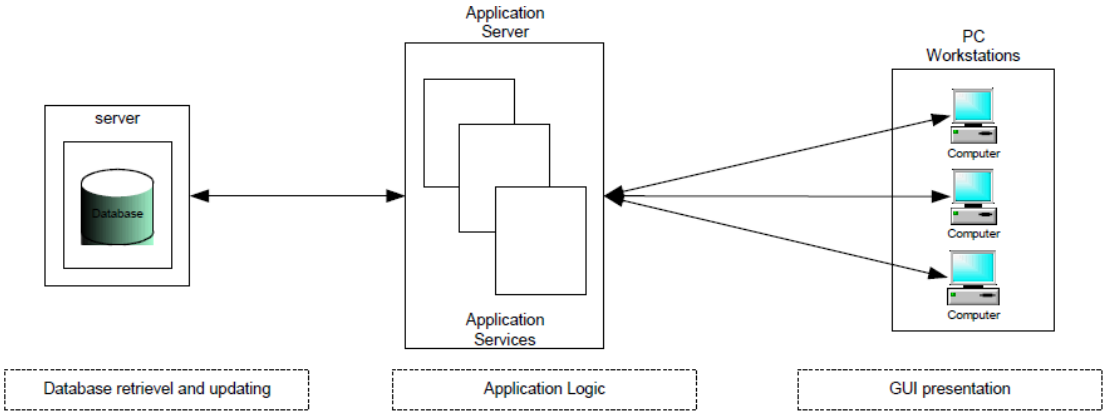


Figure 9: Three tier System Architecture.

3-tier architecture provides easy integration of data and services where each layer abstracts its services to the layer it provides services to.

4.3.1. *Presentation Layer*

Presentation layer in this application is the graphical user interface that has been developed through which user can enter input data such as search query terms, filter criterion, user credentials, shipping information, payment information, cart items and can be passed on to the application server where the business logic resides for further processing of such information. In this application this presentation layer has been designed using ASP.net Framework and Ajax framework.

Throughout the application Ajax Update panels have been extensively used to handle the post back problems of dropdown lists, hyperlink clicks, performing filtering of products, search of products etc. The left menu that displays throughout the application is an Ajax control named accordion and I have used this to display the categories and sub categories that are available in the web site. Watermark Textbox extenders have been used to let the users know about the kind of functionality that particular control is supposed to do. As an example in the home page in the right hand side we can see two text boxes with watermarks Book Title and Book Description which lets the users know that they can filter the products by entering the title and description. Such watermark extenders have been used wherever required. Collapsible Panel extenders have also been used to minimize the space on the screen so that users can expand the panel only if they

have the need to view that section. During the checkout process a user can add receiver details by the click of a button which expands the section on demand. By doing this we are giving a clean look to the screen before we add the details. If we want to hide the details we can simply click the same button. Filtered Textbox extenders have also been used where ever user is required to perform data entry. As an example a text box which accepts a phone number can be made to accept only numeric values by the use of filtered textbox extender so that user doesn't enter any alphabets in that field by mistake. By the use of such controls a lot of coding can be reduced since we are not writing any code to handle any kind of validations for such fields. Modal popup extenders are used in product details page to pop up a screen which can display more information about the product. Auto populate extender has been used to auto populate book names as and when the user starts typing in a book name in the search box. In normal C# code to implement such kind of functionality a lot of Java script code must be written but use of auto populate extenders simplifies this process because all we need to handle is the question of populating the requested items from the database. Major advantage of using all the above Ajax controls is the ability to have less code written and yet more functional look to the application. All these controls do not cause any post backs to the server and interaction with the database if any will be done without the raise of any external event.

4.3.2. *Business Logic Layer*

This is where the heart of the application functionality resides. All the modules that have been explained in section 4.3 are responsible for carrying out the business logic of this application. Class files as well as web services along with external dll's have been used to implement this layer.

4.3.3. *Data Access Layer*

This is the layer which handles the database requests from the Business logic layer and responds back with the results to the BLL which then displays them using the presentation layer on the GUI for the user. MS Sql server 2005 is the database that has been used in this layer for this application. Usage of stored procedures and triggers has made access to this layer easy without much coding at the business layer.

5. IMPLEMENTATION

5.1. Database Design and Implementation

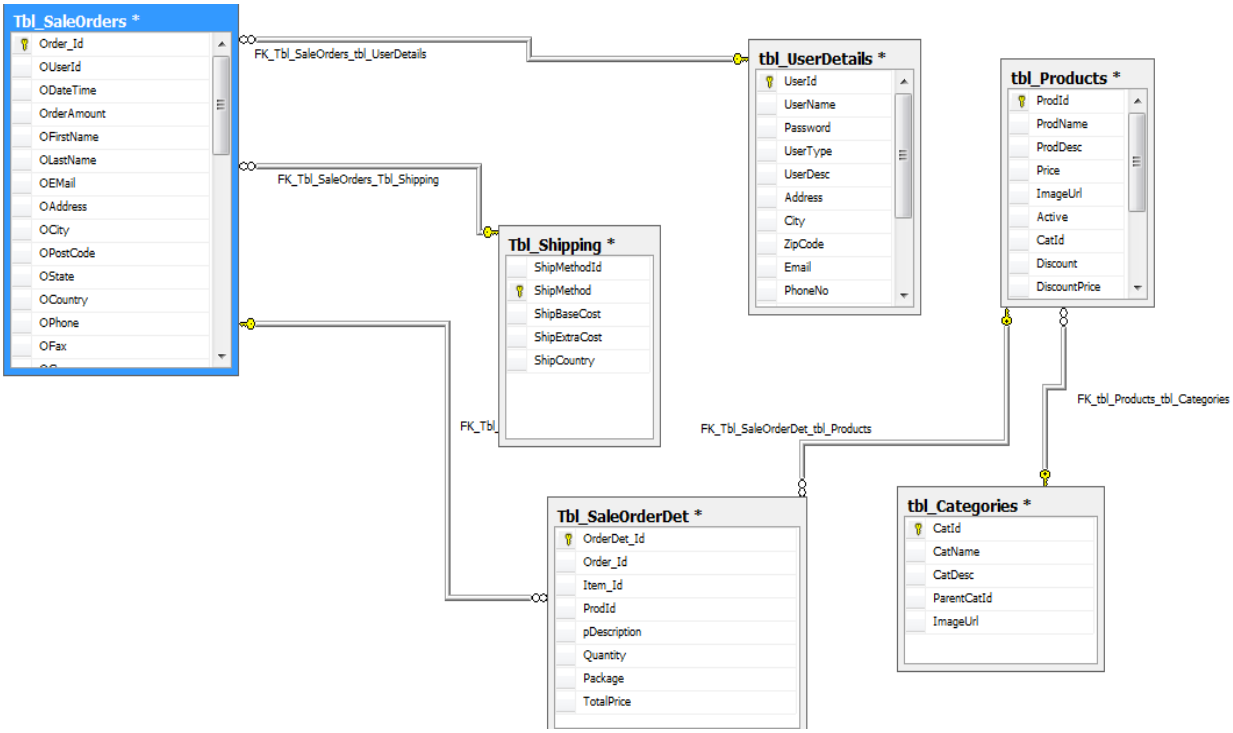


Figure 10: Database Implementation with MS Sql server 2005.

Above diagram represents the schema of this application. Along with these tables there are stored procedures and triggers developed for this application.

5.2. User Interface Design and Implementation

User interface has been designed in Asp.net 3.5 with Ajax 3.0 Framework. Ajax controls like Update panels, Accordion, Watermark Extenders, FilteredTextBox Extenders, collapsible panel extenders have been majorly used throughout this application. Asp.Net controls like Datalists and Gridview have been extensively used.

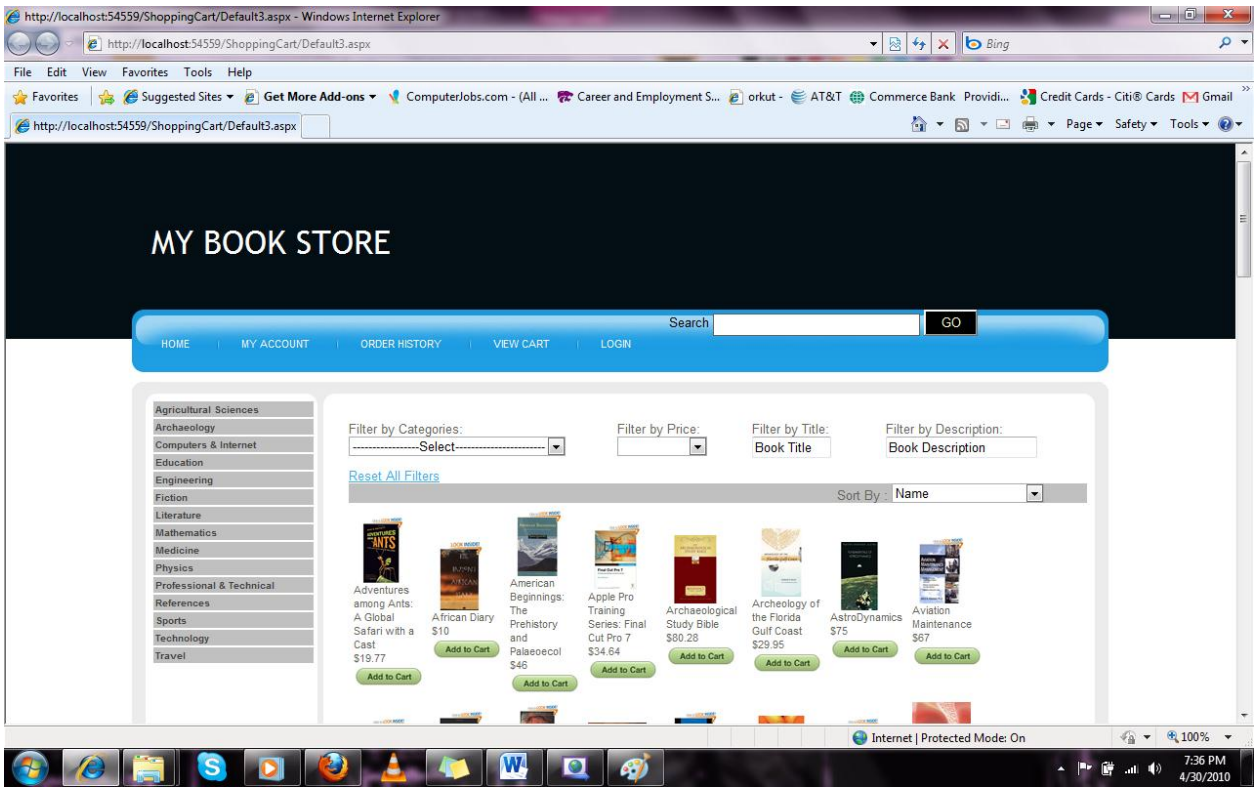


Figure 11: Home Page displaying all Available Products.

Above screenshot is the home page of the web site. All Available products are displayed here. Left menu displays all the available categories in the website and each category has a sub category below it. Right panel shows the filtering options available for the user to filter the available products. On top of the screen we have a search box where user can search for a book title. User has also the option of sorting the products based on Name, Price from low to high and from high to low by choosing an option from the Sort By dropdown. Filter by Title and Filter by Description have been extended to show what they are supposed to do by having the Term “Book Title” and “Book Description” on their textboxes so that user can easily understand the functionality.

Once the user adds a product into the cart he is taken to the page as shown in the next screenshot.

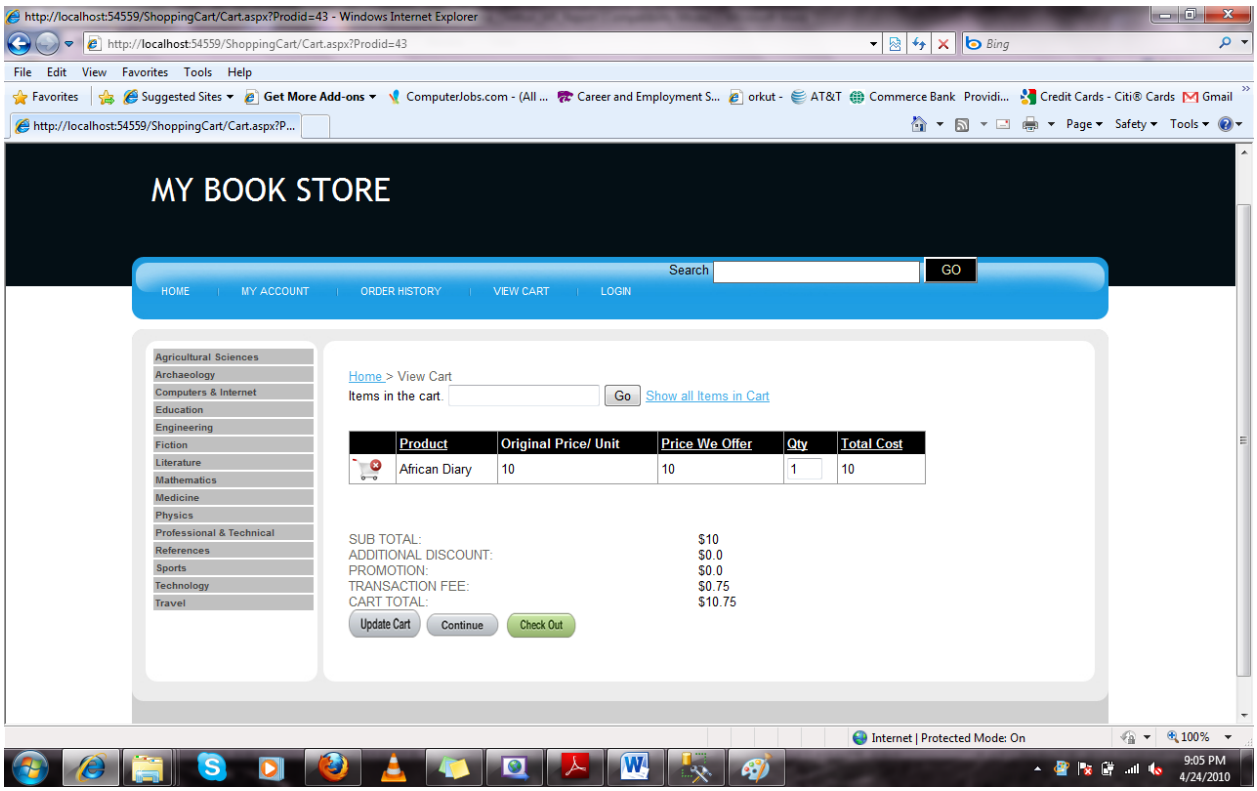


Figure 12: Shopping Cart Page

In the above screen shot we can see the products that have been added. To update a quantity we simply need to enter a number in the Qty field in the grid and click Update Cart Button. To Search for an item in the cart we can search using the text box provided above the grid and to clear a search we need to use the "Show all items in Cart" link button.

Clicking on an image in the home page takes us to the product details page which looks like

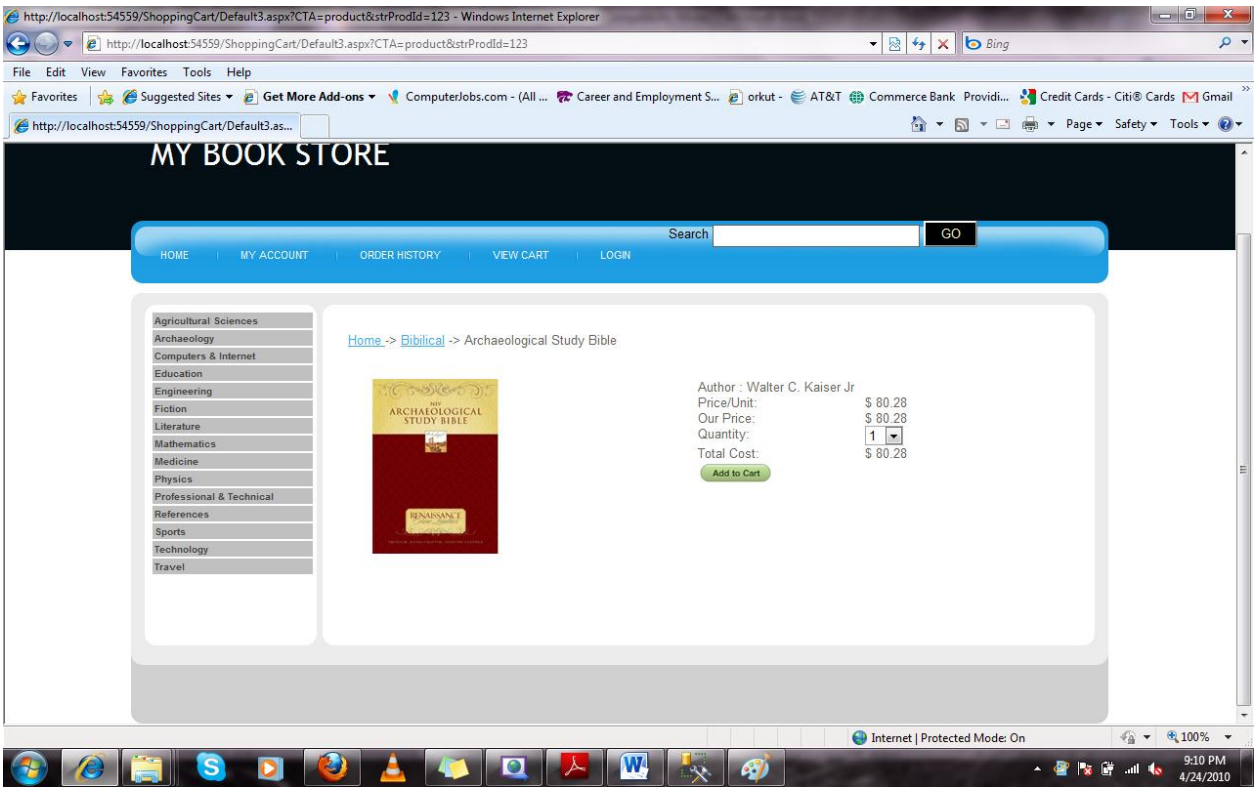


Figure 13: Product Details Page.

In the above screen we can view enlarged image of the chosen book along with other details such as the Author of the Book, price of that book and also the option to add the book from this page also. Top of this screen shows the user where he is presently at along with the option to navigate back to the origin.

Clicking on View Cart link in the top menu shows all the items that are currently in the cart. Along with items it also displays a summary about the cart total. Below is the screenshot for the same.

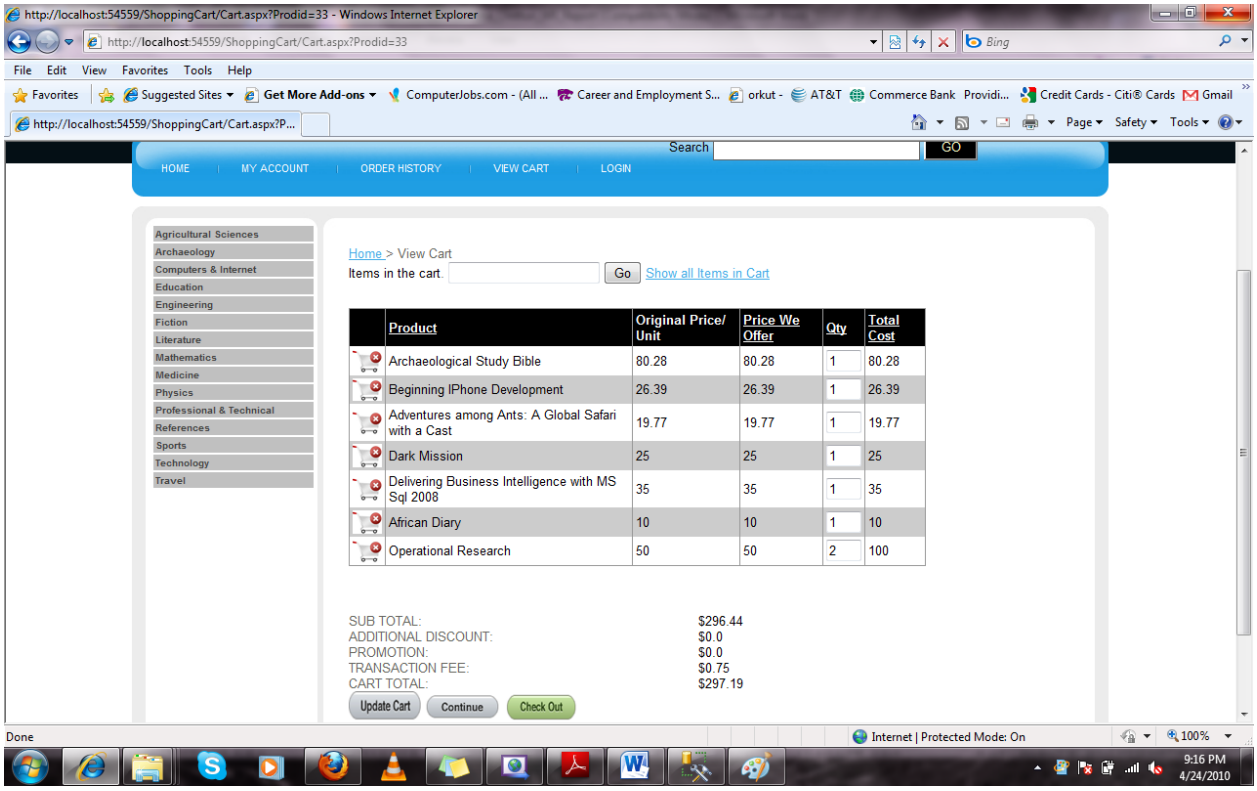


Figure 14: View Cart Page

An Order summary page would look like the below Screen Shot.

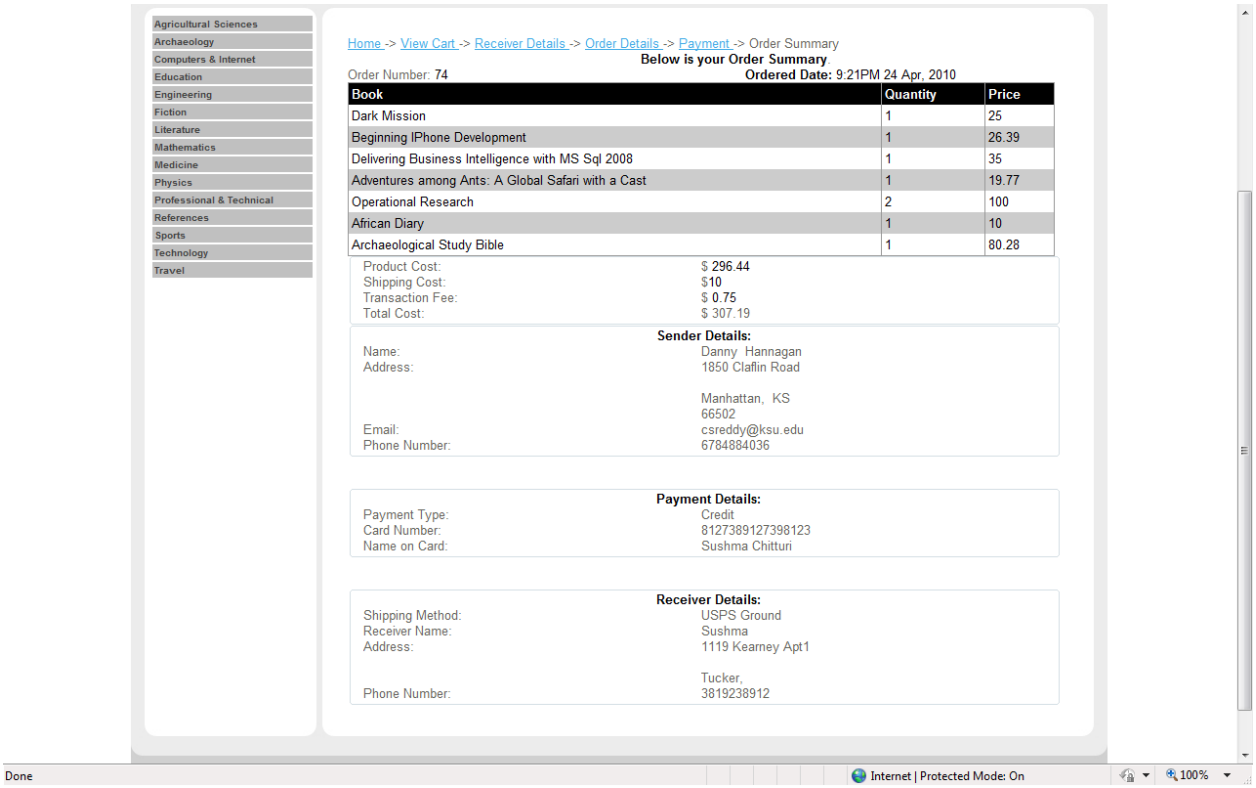


Figure 15: Order Summary Page

5.3. TECHNICAL DISCUSSIONS

The major technical aspect of this application is the sophisticated search engine. In the home page of this application all the available products are displayed which can be in huge numbers. There can be two kinds of users who could be using this application. A user who knows before-hand what book he is looking for and the other who has very little information about the book. For the former users a simple search has been enabled where a user can type in the name of the book and book starting with those names are auto populated in the dropdown and user can simply select the book his choice and view the details and checkout but for the latter users effective filtering techniques have been implemented where a user can filter the results based the department or the category of the book, price range, title of the book, description of the book etc.,

Once an item has been added to the cart user can simply update the quantity in the cart itself rather than going back to the products page and re-adding the product. He can also remove an item from the cart by the click of a delete button that is provided in the cart control itself and any update actions caused by the user automatically updates the pricing of the cart and all these updates are done at the client side itself thereby reducing the calls to the server every time an update is being made which makes this application better responsive.

Navigability is another important feature in this application which focuses on the ability for the user to easily traverse within website to any page of interest. Each page has a section which has links to other pages from where user can easily navigate and all the information in the current page will be saved in sessions before a user is redirected to other pages so that no information is lost between these traversals.

This application has also a feature where in the user can search for an item in his shopping cart and this feature becomes handy when there are numerous items in the cart and the user has to modify a particular item in the cart.

One of the other nice features this application has a zip-code finder service which is basically updating city and state information based on the zip code given by the user while updating the shipping information thereby reducing some of the data entry work done by the user. This feature has been implemented using Yahoo Maps API.

Break down of Program Code: My initial estimate of Lines of Code was about 4000. After the application has been fully developed the estimate has increased to 4534 lines which include JavaScript, Automated C# Code, Handwritten Code and CSS Code. The reason for this number is the reason that this application has administrator site as well as the user site and both these sites have been fully developed. Breakdown of the same is listed below

Code Type	Number of Lines
Java Script	50
Automated C#	1012

Hand Written C#	2387
CSS	533
SQL Stored Procedures	552

Major reason for such increase is the fact that it has both sites and also my inability to estimate the value properly during initial stages of development. A total of 250 working hours spanning over 4 months have been spent on designing and developing this application and an approximate of additional 10 working hours have been spent on testing the application for its correct functionality after the application has been fully developed.

6. TESTING

Software testing is the process of validating and verifying that a software application meets the technical requirements which are involved in its design and development. It is also used to uncover any defects/bugs that exist in the application. It assures the quality of the software.

6.1. *White Box Testing*

White box testing is one of the methods of testing where the tester has access to internal data structures and algorithms along with the code that implement the functionality of the system. White box testing ensures that

- All logical paths in the code have been executed at least once.
- All conditional statements have been executed for both true and false results.
- All loops in the code have been executed.

This kind of testing includes all kind of static testing. Each and every module of the application has been rigorously tested for both its correctness as well as the faults by giving both valid and invalid inputs and generating error and success messages. This kind of testing has been done throughout the lifecycle of the application. All the inputs that are given to the

BLL have been validated against various criterions and any non-conformity to these validation rules have been dealt with user friendly messages.

6.2. *Unit Testing*

This refers to testing functionality of each and every section of code that is written in each single class and to verify if that particular code is performing the expected functionality. This is done by the developer of the code himself simultaneously while the code is being developed. This testing is done for each and every module that has been written in the application independently.

Testing this application involved testing each and every module as and when it was finished to test and see if the required functionality has been achieved or not. At each and every stage I have tested these modules during their coding phase for its functionality until the modules have produced desired results. All the above mentioned nine modules were tested in isolation.

In the cart module which handles all the functionality of the cart tests have been carried out to make sure that if a book which is already present in the cart has been added again by the user the quantity of that book will be incremented automatically rather than showing another entry for the same book in the cart. It has also been ensured that if the quantity of an item is set to zero and if the user says update cart that particular item will be removed from the cart. While the updates are being made I made sure the pricing section updates automatically and correctly.

In the Available products module tests have been carried out to ensure that all the books that are available in the database are displayed on the home page along with their images, name of the book and its price. It has also been ensured that all categories are populated in the filter section of the screen and all price ranges too.

In login module only registered users are allowed to access the system for checking out the items in the cart and certain pages are accessed by only valid registered users. Logics have been applied here to authenticate users and allow only registered users to access those pages.

In the checkout module tests have been made to make sure that the user enters all valid and all the required information in payment section, shipping section and that only valid data is being passed onto the database.

All the above tests are carried out by using the built-in functionality of generating unit test cases for ASP.NET application in Visual Studio 2008. All the test cases run under asp.net context pull the required information from web.config file of the application automatically as and when the test cases are run. Each and every module has been tested by generating test cases through this utility.

Test cases were written to retrieve all available books from the database, login authentication, a search query term that matches the name of a book, books pertaining to a certain filter criteria. All these test cases have been successfully executed

Following are the test cases that were tested for different inputs and desired results.

1. Login method was tested so that only registered users were allowed to login into the system and proper messages were displayed if there is an error while logging in.
2. Hitting a back button doesn’t produce any undesired results in view cart page like adding a new product into the cart.
3. Entering a value zero in the quantity box in cart removes that particular item from the cart.
4. Price and quantity values are updated properly whenever user clicks Update cart button in view cart page.
5. Navigation Link buttons take users to designated pages.
6. Categories and sub categories are displayed in the left menu.
7. Proper filtering of books.
8. Filtering Panel is displayed only when there are books available.
9. All Orders are displayed when user queries his order history.

Some of the bugs that were encountered during manual testing of my application were

1. In view cart page if a user hits a back button and then forward button I observed that a product which was previously added gets added to the cart again.

   Resolution: This error was resolved by using proper session values and handling of post backs.

2. Authentication failures were encountered.

   Resolution: The reason for this was when users get registered with the site their usernames were stored in Uppercase in the database and hence I had to check with the casings of the usernames in the login screen and resolved this issue.

3. Adding a product that was already present in the cart resulted in adding a new row to the cart.

   Resolution: Logic was written to handle this issue so that a new row doesn't get added but instead the quantity of that particular product gets incremented.

4. Initially I experienced some null pointer exceptions since I was binding the data even if nothing was returned from the database.

   Resolution: Proper validation techniques were implemented to handle any null pointer exceptions throughout the application.

5. Entering alphabetical characters where numerical characters are expected resulting in failure of the application.

   Resolution: Use of filtered extenders from Ajax framework was employed to handle such situations.

6.3. *Integration Testing*

This kind of testing makes sure that the interfaces between components are verified against the design. In this application all individual modules are integrated in an iterative fashion where after each module has been unit tested individually it has been integrated with

previously tested module and the combination has been tested together for the functionality. Tests have been made to make sure that the login module has been executed before checkout module has started and administrator module has started before starting the add categories module or add sub categories module etc.

6.4. *Non Functional Testing*

Nonfunctional testing ensures that the application works correctly even when wrong inputs are given to the application. For this application as an example I entering an email address without @ symbol and checked to see if the application accepts the email address while registering a new user and the system shows an alert asking us to enter proper email address. Certain input fields in the application are allowed to accept only numerical values and I made sure that they do not accept any other values other than numerical by use of Ajax Control extenders.

6.5. *Validation Testing*

This is the final assurance for the quality, functionality and behavioral requirement of the application.

This comprises of validation test criterion which makes sure that input fields receive proper type of inputs (as an example if user is entering data into a phone number field it has to be only numbers but not characters) configuration review which is to make sure that software has been correctly configured and that the system has met the required technical specifications and Alpha testing which is testing the application at the developer's site which is primarily in my laptop.

Some of the tests that I performed to test the functionality of the application are:

- Only registered users are allowed to log-in to the system.
- Addition of a new product adds a new row to the system.
- Updating the quantity of a product automatically updates the cart total.

- Addition of an existing product in the cart updates the quantity field of that product in the field.
- An Administrator has access both to the administrator section as well as the main application.
- Atomicity of transactions is ensured by having separate sessions created for each user.
- All the details about the book are correctly being displayed.
- Filtering of all criterions produce desirable results.
- User being able to navigate to the previous page without loss of information.

6.6. *Performance Testing*

Performance testing is performed to determine how fast a system performs under a particular load. It is also used to validate and verify other attributes of the system such as scalability, reliability and resource usage. Load testing is primarily concerned with testing that can continue to operate under specific load be it large amount of data or be it numerous users.

To perform load testing I have used Apache JMeter which can be used to test performance both on static as well as dynamic resources. This simulates a heavy load on the server, network or object to test its strength or to analyze overall performance under different load types.

I have performed load testing on pages such as the home page, shopping cart page, Login page and all these tests have been performed with the application and load testing tool on the same server.

Major factors that I have chosen to test the stress and performance levels of this application were majorly the number of users accessing the system. I have tested the application for constant load as well as under peak load conditions to see how well it behaves.

6.6.1. *Testing Configuration*

Testing in Different environments leads to different performance results. An Application's performance especially a web application's performance depends on the configuration of the server that it is hosted on, the speed of the RAM. Speed of the Internet also determines how good an application is performing. Results may vary when we have our application as well as the databases where we make requests are on the same server as both of them would be competing for resources of a single machine. If the application and database are on different machines then performance may vary as there would be less competition for the resources.

Testing environment for this application is a Windows 7 professional edition laptop with a 2GB RAM and 2.2 GHz of CPU speed. This configuration meets the hardware specifications which were specified at the very beginning of the document. I have tested my application on my laptop which acts as the local web server which has both the application and its database on the same server and hence these results do not include factors like bandwidth usage.

6.6.2. *Results*

Tests were conducted to interpret the application's performance when the load on the application increases. To achieve this interface I have simulated the application to get concurrent requests from 50, 100 and 500 users and also have certain parameters that we need to set before running a test. Those parameters are namely Ramp-Up period and Loop Count. Ramp-Up period determines how long to delay between starting each user. The value that was set in this testing was 5 and the other parameter that we need to set is the loop count which determines how many times we need to repeat the test. I used in a value of 500 for my testing. I have simulated 50, 100 and 500 users and repeated the test.

Users	**Ramp-Up Period**	**Loop Count**	**Home Page Average Response (ms)**
50	5	500	995
100	5	500	996
500	5	500	1010

From the above results it can be seen that the average response of the home page increases as the number of users increase but it doesn't deteriorate considerably.

Interpreting the Results:

Response time of an application increases as the number of users increase but the increase is very minimal since we have the loop count and ramp-up period constant. By keeping the ramp-up period constant and increasing the number of users we are increasing the delay between the requests which means the number of requests handled by the server per second increases and hence the longer response time.

Comparing Response Times of Two Web Pages

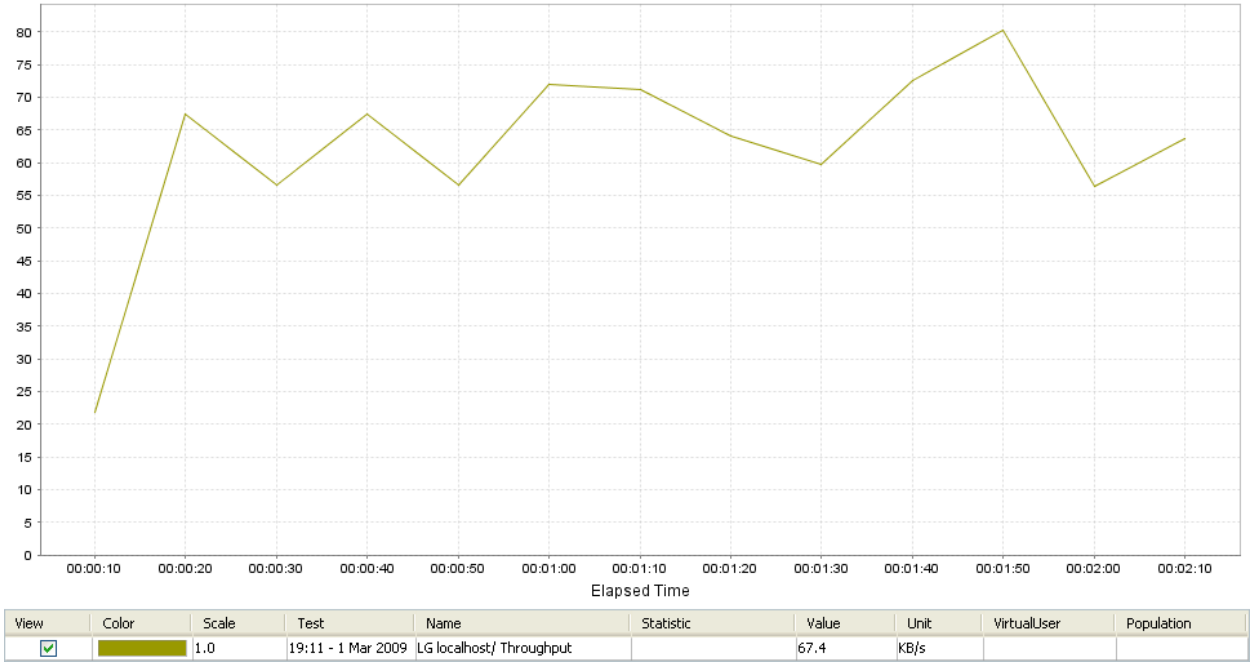
Tests have been performed to see how two pages with different functionality behave when the numbers of users are increased but the loop count and ramp-up period are kept constant with values of 150 and 10sec respectively. Home page is more complex in its business logic when compared to product details page which is a simple information retrieval from the database.

Users	**Home Page (ms)**	**Product details Page (ms)**
100	28000	1060
500	105300	8000
1000	135000	11000

Interpreting the Results:

A Page which has more business logic involved has a longer response time when compared to a simple page because there would be more number of sockets opened up to the database by many users at a single point of time and hence the application can get more resource hungry and thus there would be obvious delay in response. The reason for this is the complex calls to the database in handling the functionality.

Below are the graphical representations of the throughputs for both a simple web page and a complex web page when requested by 10 users simultaneously under peak load situation. To test this situation I have made use of NeoLoad tool developed by NeoTys where we can simulate any number of virtual users and assigning the pages in the website to these users and setting configuration values such as peak load, constant load and ramp-up.



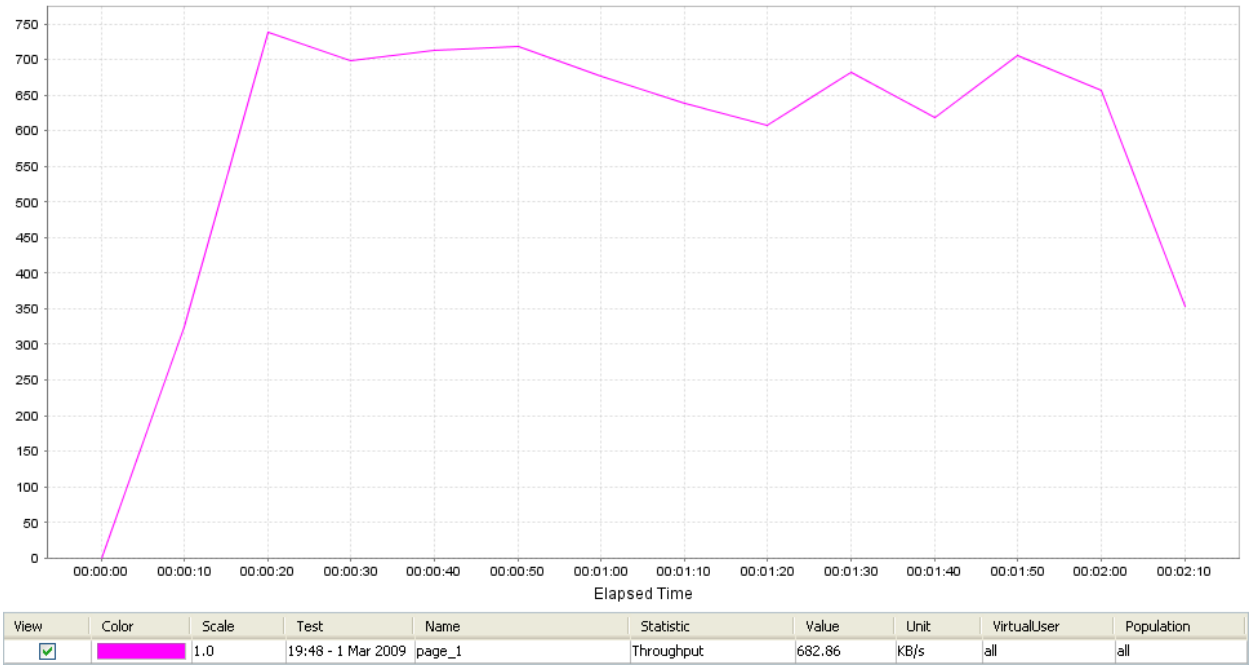


Figure 16: Throughput for a simple web page. (Product Details Page)

Figure 17: Throughput for a complex web page (Home Page)

Top 5 Maximum Response Time for both Constant and Ramp-Up Loads have been displayed below

Constant load:

Page	Duration
page_1	0.453
page_4	0.438
page_5	0.406
page_3	0.344
page_6	0.343

Ramp-up:

Page	Duration
page_6	0.188
page_4	0.188
page_2	0.187
page_3	0.172
page_1	0.171

Interpreting the Results:

Tests under constant and ramp-up have shown that throughput under constant load conditions is high when compared with continuously increasing loads. The reason for this is because the application and database are located on the same server we have competition for allocation of resources. This is a Bottleneck situation for the application and the solution to overcome this situation would be to increase the CPU Speed and Memory to handle increasing load situations.

Apart from using Apache JMeter as well as NeoLoad I have made use of Microsoft's Web application stress tool to simulate my system with hundred users for about an hour with a constant load on a low level network bandwidth. In order to simulate the above script I set stress level to hundred which means there would be hundred threads that would be executing simultaneously and increased the stress multiplier factor which denotes how many sockets have

been opened up for each thread and stress level multiplied by number of threads denotes the number of users hitting the application. Other factors such as the amount of time the test has to run along with request delay between users and bandwidth have been set and I have initially tested this application for an hour with request delay set to 5 seconds.

With these parameters set in WAS tool I tested and I observed that CPU utilization was not close to full strength as it is supposed to be. Summary report has shown that only 40% of CPU Utilization. I have then increased the number of users to 200 and then tested with rest all parameters remaining the same. Now I have observed that my CPU Utilization has increased considerably to about 75%.

In order to achieve 100% utilization may be the time of testing the script should be increased to more than an hour because the longer the application runs, the waiting time of resources gets increased and hence the system gets bogged down. In testing my application major bottleneck was CPU since 100% utilization was encountered very few times during the test run.

From above statistics it can be said that the application can handle normal load of about 100 users with fast response times so that users need not wait long to perform an action.

7. RESULTS AND CHALLENGES

For now this has been designed as on online cart primarily for books but with simple changes this can be extended to any e-commerce application since this has all the features that a user looks for in a shopping cart like the feature filtering products on various criterion, easy navigability throughout the site, very simple yet effective method of adding and updating the cart and also the ability to search for items within the cart.

7.1. Challenges

Major challenges that I have encountered while developing and designing this application are:

- Effective designing of web pages so that they are rendered properly in both IE and Mozilla Firefox.
- Designing of database so that there is no redundancy. Normalization of the tables that have been designed.

- Learning of Ajax framework for implementing many controls in the application and the usage of java script for client side validations.
- Using Web services in the application.

8. CONCLUSIONS

MyBookStore an online shopping system for books has been designed and developed to serve the purpose of a web application that is fast and user friendly. This has various features which would be of primary interest to the users such as filtering the books on various criterions, auto populating of books that are available as an when a user starts to type in a book name, easily manageable shopping cart, able to search for items in the cart and easy navigability. This one also has an administrator side which handles the responsibility of managing the website. It has been developed using AJAX framework which gives a neat and clean UI which is a flicker free one because of the post backs that are caused.

8.1. *Limitations*

Some of the limitations of this application are

- Validating the check-out process which means validating the credit card details that have been entered by the user while checking out.
- Ability for the users to save their cart so that they can check out later.
- Ability for the users to subscribe to price alerts for books.

8.2. *Future Work*

Current system can be extended to allow

- Drag and drop of the items into the cart.
- Saving the cart for future check-out purpose.
- Subscribing for price alerts.
- Ability to save the billing and payment details so that data entry work for a user can be reduced while checking out.
- Ability to add multiple receivers.

9. REFERENCES

- "Microsoft Ajax Control Tool Kit", Microsoft Corporation 2009 (http://www.asp.net/ajaxlibrary/act_tutorials.ashx) visited on 12/02/2010.
- "Software Testing", Wikipedia, the free encyclopedia (http://en.wikipedia.org/wiki/Software_testing#Testing_methods) visited on 24/04/2010.
- "Samples in Report Writing", K-State Research and Exchange (http://krex.k-state.edu/dspace/handle/2097/1287) visited on 01/04/2010.
- "Open Source E-Commerce Solution", nopCommerce 2010 (http://www.nopcommerce.com/) visited on 10/02/2010.
- "Apache JMeter", Apache Software Foundation 1999-2009 (http://jakarta.apache.org/jmeter/) visited on 10/04/2010.
- "Smart Draw", SmartDraw.com 2010 (http://www.smartdraw.com/) visited on 21/04/2010.
- "Nhibernate for .Net", Red Hat Middleware, 2009 (https://www.hibernate.org/343.html) visited on 10/03/2010.
- "Nhibernate - Relational Persistence for Idiomatic .NET", hibernate.org (https://www.hibernate.org/hib_docs/nhibernate/html/quickstart.html) visited on 10/03/2010.
- Geoffrey Sparks, Database Modeling in UML, Sparx Systems.
- "A UML Profile for Data Modeling", Scott.W.Ambler, 2009 (http://www.agiledata.org/essays/umlDataModelingProfile.html) visited on 21/03/2010.
- "How to: Measure Asp.Net Responsiveness with Web Application Stress Tool", Microsoft Support 2010, (http://support.microsoft.com/kb/815161) visited on 26/04/2010.