E-COMMERCE ONLINE MUSIC STORE

by

RESHMA D SAWANT

Bachelor of Engineering (B.E), Vidyavardhini’s CoE, University of Mumbai, Mumbai, INDIA. 2004

A REPORT

submitted in partial fulfillment of the requirements for the degree of

MASTER OF SOFTWARE ENGINEERING

Department of Computing and Information Sciences
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas
2008

Approved by:

Dr. Daniel Andresen
[Major Professor]
ABSTRACT

The objective of this project is to design a Music store web application with user interface which will enable them to browse, search, get song recommendations and buy the song-items of their choice. The motivation of this project comes from my desire to learn the increasingly growing field of .NET, SQL server database designing, website designing and their growing popularity by taking up this case study.

The word “design” in the context of a Web Application can mean many things. Its most popular usage probably refers to the visual and user interface (UI) design of a web site. This aspect is crucial because, the visitor is often more impressed with how a website looks and how easy it is to use than about which technologies and techniques are used behind the scenes, or what operating system the web server is running. If the site is hard to use and easy to forget, it just doesn’t matter what technologies was used to create it. Unfortunately, this truth makes many inexperienced programmers underestimate the importance of the way the invisible part of the site is implemented—the code, the database, and so on. The visual part of a site gets visitors interested to begin with, but its functionality makes them come back. A web site can sometimes be implemented very quickly based on certain initial requirements, but if not properly architected, it can become difficult, if not impossible, to change.

Thus, performance is also a major thrust area in the Web application which is one of the main reasons why users get attracted to it. Growing user needs should be taken in to concern with new features to be included. Effective performance can be achieved by making proper database design strategy. Also, easy navigation also needs to be accomplished while executing this project.
# TABLE OF CONTENTS

Chapter 1 – Vision Document .......................................................................................... 9  
1. Introduction ................................................................................................................ 9  
2. Project Overview ...................................................................................................... 10  
  2.1 Introduction ................................................................................................... 10  
  2.2 Goal ............................................................................................................... 14  
  2.3 Purpose .......................................................................................................... 24  
3. Requirements Specification...................................................................................... 14  
  3.1 Main Requirements ....................................................................................... 14  
  3.2 External Interfaces Requirements ................................................................. 15  
  3.3 Critical Use Case Requirements .................................................................... 16  
  3.4 Performance Requirements ........................................................................... 26  
  3.5 Assumptions .................................................................................................. 26  
  3.6 Constraints ..................................................................................................... 27  
  3.7 Environment .................................................................................................. 27  
Chapter 2 – Project Plan ................................................................................................. 28  
1. Task Breakdown ....................................................................................................... 28  
  1.1 Inception Phase ............................................................................................. 28  
  1.2 Elaboration Phase .......................................................................................... 28  
  1.3 Production Phase ........................................................................................... 29  
2. Architecture Elaboration Plan .................................................................................. 31  
  2.1 Revision of Vision Document ....................................................................... 31  
  2.2 Revision of Project Plan ................................................................................ 31  
  2.3 Formal Requirements Specification .............................................................. 31  
  2.4 Architecture Design....................................................................................... 32  
  2.5 Test Plan ........................................................................................................ 32  
  2.6 Formal Technical Inspections ....................................................................... 32  
  2.7 Architecture Prototype .................................................................................. 32  
3. Cost Estimate............................................................................................................ 33  
4. Implementation Plan ................................................................................................ 34  
  4.1 Deliverables ................................................................................................... 34  
  4.2 Work Breakdown Structure ........................................................................... 35  
Chapter 3 – Architecture Design .................................................................................... 37  
1. Introduction .............................................................................................................. 37  
2. Architecture of the System ....................................................................................... 37  
  2.1 Presentation Tier ........................................................................................... 38  
  2.2 Business-Tier ................................................................................................. 40  
  2.3 Data Tier (Database) ..................................................................................... 52  
3. Formal Specification ................................................................................................ 53  
Chapter 4 – Inspection List ............................................................................................ 64  
1. Purpose ..................................................................................................................... 64  
2. Items to be Inspected ................................................................................................ 64  
3. Formal Technical Inspectors .................................................................................... 64  
4. Formal Technical Inspection Checklist .................................................................... 64  
Chapter 5 – Component Design ..................................................................................... 66
2.4  Software Set-up .................................................................................................. 115
3.  OnlineMusic Store Usage ..................................................................................... 119
  3.1  User .................................................................................................................. 119
  3.2  Administrator ................................................................................................. 129
Chapter 10 – Project Evaluation ................................................................................. 160
  1.  Introduction ........................................................................................................ 136
  2.  Problems Encountered ....................................................................................... 136
  2.1  Solution domain research and learning ...................................................... 136
  2.2  Learning ASP.NET and C# Language ....................................................... 136
  2.3  Security Issues ................................................................................................ 137
  3.  Source Lines of Code ........................................................................................ 138
  4.  Project Duration .................................................................................................. 141
  5.  Lessons Learnt .................................................................................................... 142
References .................................................................................................................. 143
Appendix A. Srunokshi Neelakantan’s Formal Inspection Checklist ......................... 167
Appendix B. Inspection Letter .................................................................................... 169
  1.  Srunokshi Neelakantan’s Inspection Letter ..................................................... 145
LIST OF FIGURES

Figure 1 Page Flow Diagram ................................................................. 13
Figure 2 Overview of System .............................................................. 13
Figure 3 System in Three-Tier Architecture ....................................... 15
Figure 4 Use Case Diagram ................................................................. 16
Figure 5 The Project Gantt Chart ......................................................... 34
Figure 6 Three-Tier Architecture Diagram .......................................... 38
Figure 7 Domain Model Diagram ....................................................... 40
Figure 8 Page Flow Diagram of User .................................................. 41
Figure 9 Page Flow Diagram of Administrator ................................. 41
Figure 10 User Class Diagram ............................................................ 41
Figure 11 Customer Class Diagram .................................................... 42
Figure 12 Administrator Class Diagram ............................................ 43
Figure 13 SessionManager Class Diagram ......................................... 43
Figure 14 Genre Class Diagram ......................................................... 44
Figure 15 Product Class Diagram ....................................................... 44
Figure 16 ShoppingCart Class Diagram .............................................. 45
Figure 17 CartItem Class Diagram ..................................................... 46
Figure 18 Order Class Diagram ......................................................... 46
Figure 19 OrderDetail Class Diagram ................................................ 47
Figure 20 ShippingInfo Class Diagram .............................................. 47
Figure 21 keyword Class Diagram ................................................... 47
Figure 22 searchFacade Class Diagram ............................................. 48
Figure 23 RecommendationSet Class Diagram ................................. 48
Figure 24 productSet Class Diagram ................................................ 48
Figure 25 System Login Sequence Diagram ....................................... 49
Figure 26 System Buying Sequence Diagram ..................................... 50
Figure 27 System Search Sequence Diagram ..................................... 51
Figure 28 Physical Database Schema ............................................... 52
Figure 29 Middle Tier Class Diagram ............................................... 66
Figure 30 User Class Component Diagram ....................................... 67
Figure 31 Customer Class Component Diagram ............................... 67
Figure 32 Administrator Class Component Diagram ............................ 68
Figure 33 SessionManager Class Component Diagram ...................... 69
Figure 34 Genre Class Component Diagram ..................................... 69
Figure 35 Product Class Component Diagram .................................... 70
Figure 36 ShoppingCart Class Component Diagram .......................... 71
Figure 37 CartItem Component Diagram .......................................... 72
Figure 38 Order Class Component Diagram ..................................... 72
Figure 39 OrderDetail Class Component Diagram ............................. 73
Figure 40 ShippingInfo Class Component Diagram ............................ 73
Figure 41 keyword Class Component Diagram ................................... 74
Figure 42 searchFacade Class Component Diagram ........................... 74
Figure 43 RecommendationSet Class Component Diagram ............... 75
Figure 44 productSet Class Component Diagram .............................. 75
Figure 69 Breakdown for Each Phase ................................................................. 129
Figure 70 Phase I Breakdown ........................................................................... 129
Figure 71 Phase II Breakdown ........................................................................... 130
Figure 72 Phase II Breakdown ........................................................................... 130
LIST OF TABLES

Table 1 The Work Breakdown Structure Table ......................................................... 36
Table 2 Client Site ASP.NET Web Forms .............................................................. 39
Table 3 Administrator Site ASP.NET Web Forms .................................................. 39
Table 4 Database Table ......................................................................................... 52
Table 5 Formal Technical Inspection List ............................................................ 64
Table 6 Features to be Tested ............................................................................... 84
Table 7 Test Case Result Summary ...................................................................... 95
Table 8 Register Test Case Result Summary ....................................................... 96
Table 9 User Login Test Case Result Summary .................................................. 96
Table 10 Search Test Case Result Summary ....................................................... 97
Table 11 Edit Shopping Cart Test Case Result Summary .................................... 97
Table 12 Place Order Test Case Result summary ............................................... 98
Table 13 Administrator Login Test Case Result Summary .................................. 98
Table 14 Create & Delete Product Test Case Result Summary ......................... 99
Table 15 Create & Delete Category Test Case Result Summary ......................... 99
Table 16 Create & Delete Genre Test Case Result Summary ............................. 100
Table 17 Manage Orders Test Case Result Summary ........................................... 100
Table 18 Manage Shopping Carts Test Case Result Summary ............................. 100
Table 19 Project Duration .................................................................................... 138
Table 16 Activities Breakdown in Each Phase .................................................... 138
ACKNOWLEDGEMENTS

Sincere appreciation and warmest thanks are extended to the many individuals who in their own ways have inspired me in the completion of this project.

First, I want to thank Professor, Dr. William Hankley who persuaded and continuously guided me during the whole duration of my project. His patience and expert professional guidance kept me on track in this professional endeavor.

I wish to acknowledge my Major Professor, Dr. Daniel Andresen and committee member, Dr. Torben Amtoft, for their assistance and insightful comments, and who willingly shared their expertise with me.

Last but not the least; I want to express my sincerest gratitude to the two most important people in my life. To my parents, who opened my door of opportunity in education. I thank my mother, Mrs. Deepali Sawant, for all her love, and support as I continuously worked to achieve my professional goal. I thank my father, Mr. Dilip Sawant, who never failed to believe in me. Thank you for helping me discover my potentials. Your love, dedication, and commitment made this professional journey in my life worthwhile.
1. Introduction

1.1. Purpose and Motivation

The objective of this project is to implement a Music CD store web application with user interface. The motivation of this project comes from my desire to learn the increasingly growing field of .NET, SQL server database designing, website designing and their growing popularity by taking up this case study.

The word “design” in the context of a Web Application can mean many things. Its most popular usage probably refers to the visual and user interface (UI) design of a web site. This aspect is crucial because, the visitor is often more impressed with how a website looks and how easy it is to use than about which technologies and techniques are used behind the scenes, or what operating system the web server is running. If the site is hard to use and easy to forget, it just doesn’t matter what technologies was used to create it. Unfortunately, this truth makes many inexperienced programmers underestimate the importance of the way the invisible part of the site is implemented—the code, the database, and so on. The visual part of a site gets visitors interested to begin with, but its functionality makes them come back. A web site can sometimes be implemented very quickly based on certain initial requirements, but if not properly architected, it can become difficult, if not impossible, to change.

Thus, performance is also a major thrust area in the Web application which is one of the main reasons why users get attracted to it. Growing user needs should be taken in to concern with new features to be included. Effective performance can be achieved by making proper Database design strategy. Also, easy navigation also needs to be accomplished while executing this project. These are the main motivations for the project.

2. Project Overview

2.1. Introduction

This application, called the Music Store is implemented using C# and ASP.NET 2.0.
This project covers the following implementations:

1) *An online product catalog that can be browsed:* The work starts with adding many new product catalog features which includes displaying categories, products, and product details.

2) *Searching the Catalog:* For the visual part, a text box is used in which the visitor can enter one or more words to search through the product catalog. In Music CD Shop, the words entered by the visitor are searched for in the products’ names and descriptions. Also, the user can search for a particular song by entering the title, artist, style, format and the price range.

3) *A Custom Shopping Cart and checkout in ASP.NET:* A custom shopping basket is implemented, which stores its data into the local database. Also a “shopping cart summary control” is created that shows up in every catalog page except the shopping cart page.

4) *Handling Customer Accounts:* In customer account system, details such as credit card numbers are stored in a database so that customers don’t have to retype this information each time they place an order. Customers can log in via a login page or dialog box to get access to secured areas of the web site. Once logged in, the Web Application remembers the customer until the customer logs out (either manually via a Log Out button or automatically, if the session times out or a server error occurs). All secure pages in a Web Application need to check whether a customer is logged in before allowing access.

5) *Making Song Recommendations:* One of the most important advantages of an online store is the capability to customize the web site for each visitor based on his or her preferences, or based on data gathered from other visitors with similar preferences. In product recommendations system, additional products are suggested to an individual visitor in a clever way to increase sales. Here, song (product) will have recommendations based on the users’ past purchases and based on data gathered from other users with similar preferences.
6) **Catalog Administration:** This administrative interface is implemented for easy management of the web store data. The catalog administration page allows the administrator to:

- Add or remove genres, and update the details of existing genres
- View and manage the categories that belong to a genre
- Manage the list of products in a specific category, and edit product details
- Assign an existing product to an additional, or move it to another category
- Remove a product from a category or delete the product from the catalog
- Manage orders by updating their status
- Manage the shopping carts by removing those which haven’t been updated by the customer in certain amount of time.
- The administration page also needs to ask for a username and password, so that only the website administrator is allowed to perform administrative tasks.

The diagram below captures the page flow (ASP.NET Web forms) model for the user:
Figure below shows the high-level layout diagram of the system.
The said system will interact as follows:

a. The client sends request and .NET form inputs over the network.
b. The Internet Information Server (IIS) receives .Net form inputs.
c. IIS Web server
   i. Processes inputs
   ii. If required, queries to the database and retrieves data
d. The IIS Web server sends back processed output over the network as a Web page.
e. The client receives the output as a Web page.

2.2. Goal

The goal of this project is to provide an automated .NET Web application that allows a user to browse and buy a music CD over the Internet at any time.

2.3. Purpose

The purpose of this project is to explore the capabilities of the Microsoft .NET Framework and to provide a convenient service of buying a music CDs to online customers.

3. Requirement Specifications

3.1. Main Requirements

1. Construct a system with three-tier architecture. Figure 3 shows the system on three-tier architecture.
2. Microsoft Visual Studio 2005, ASP.NET, C# language, Microsoft SQL Server 2005 and XML will be used to develop the system.
3. The final product will be run on Internet Information Server (IIS).
4. Mozilla Firefox will be used as a main browser.
3.2. **External Interface Requirements**

All user interfaces are ASP.NET-generated Web pages. In order to access the system, the user will need to use a workstation with Internet accessibility equipped with Internet Explorer. It is also a must that the Microsoft .NET Framework is installed on the same machine. A broadband connection is recommended to boost the performance. A Web page will be displayed according to the user’s choice. The Administrator can add new or update the quantity and details of existing Genres/Categories/Products. The online customer can browse/search through the catalog and buy the music CDs/DVDs of his choice.
3.3. Critical Use Case Requirements

There are two actors for Online Music CD Store - a Customer and an Administrator.

A *customer* can use the system in four scenarios:

- Browse Catalog
- Search Catalog
Get Song Recommendation
Manage Account
Place Order

1) **Use Case 1: Browse Catalog Requirements:**

- **Purpose:** A customer can browse through the different genres, categories and can also view the details of the products such as the description, price and songs listing etc. Depending upon the genre/category selected, the contents of the table are accessed using a select query.

- **Input:** The User will select one of the genre and its category.

- **Output:** The system will display product list and information of the selected genre and/or category. The product list will be displayed on product.aspx page and 6 products will be displayed on each page and the rest (if any) will be on the next page. This will be executed using the “pagination” property i.e. there will a link named “Previous” and “Next” on the bottom of every product.aspx page to enable the customers to go to the next and previous pages to view products. The current page of the customer will also be displayed on every page.
2) **Use Case 2 : Perform Product Search Requirements:**

- **Purpose:** The purpose of this part of the application is to enable the customer to find the available product (music CD/DVD etc) of his choice without browsing the entire catalog.

- **Input:** The customer will hit the Search button on the top of every page. This will redirect the user to Search.aspx Web page where he/she will have the options to enter name of the music-album/CD, artist, style, format and price range of his/her choice. Also, the user can enter any text in the search text box and can choose for the system to search for all the words he entered and hit the “Search” button. This will redirect the user to the page which will display all the matched items; otherwise an appropriate message will be displayed.

- **Output:** If the user inputs are not valid (i.e. the user did not enter any of the required options), an appropriate error message will be displayed. If the inputs are valid, a message will be displayed affirming the user’s choices along with the appropriate product(s) information for the particular search. If there are no matches, the system will display an appropriate message.
3) Use Case 3: Get Song Recommendations:

- **Purpose**: The purpose of this part of the application is to enable the customer to find recommendations for the songs of his choice.
- **Input**: The customer will hit the “Get Recommendations” button on the top of every page. This will redirect the user to Recommendations.aspx Web page where he/she will have the options to enter title of the song, artist and style of his/her choice and click the “Get Similar Songs” button.
- **Output**: Initially, a list of most similar songs (to the input song) will be displayed to the user, from which the user can choose some songs and get their most recommended songs.

To implement this functionality, a method called Case-based Reasoning is used. A description of the method is included at the end of this chapter.

4) Use Case 4: Manage Account Requirements:

- **System Login**:
  - **Purpose**: This is implemented to enable user authentication. A valid user account must be used for an existing customer.
  - **Input**: The customer can login to the e-Commerce shopping system by entering his user name and password.
  - **Output**: The system will verify that the login name matches the login password. If the user name or password is invalid, the appropriate error
message will be indicated and the user will be requested to re-enter user name and password. If the user inputs are valid, the main page will be displayed.

- **System Register:**
  - **Purpose:** This is implemented to enable a new user authentication. A valid user account must be used for an existing customer or a new customer can register.
  - **Input:** If the customer is a new user, he can request to register with the system.
  - **Output:** The system displays a registration page and asks the customer to choose a user name, password and enter a valid email id, security question and answer.
5) **Use Case 5 : Manage Profile Requirements:**

- **Purpose:** User can edit, update and save his personal information.
- **PreCondition:** The user must be logged into his account to Update Personal Information. The user inputs will be saved to the database.
- **Input:** The customer can request to update their customer info.

The user will enter personal information such as:

- First name and last name
- Street address
- city, state, zip code, country
- Telephone
- Email
- Credit card information
- Billing and Shipping addresses

After entering all the information the user must click the update/save button.

- **Output:** The customer updates the customer information and the system will store the updated customer info in the system database.
6) Use Case 6: Place Order Requirements:

- **Add to cart:**
  - **Purpose:** This is implemented to add products to shopping cart while searching or browsing catalog.
  - **PreCondition:** The user must be logged in to add a product to the cart. The product will be added to a shopping cart table in the database using insert command.
  - **Input:** When the customer finds the products he wants, he adds them to the shopping cart by clicking on the “Add to Cart” button.
  - **Output:** The product will be added to the shopping cart and the system will store and keep track the information of the products that have been added into shopping cart.
- **View Cart Details:**
  - **Purpose:** This is to view contents of the shopping cart while searching or browsing the catalog. The contents of the shopping cart table will be displayed by using a select query.
  - **PreCondition:** The user must be logged in and must have at least one Cart item to view details of the shopping cart.
  - **Input:** The customer can request to view the contents of the shopping cart by clicking on the “view details” button.
  - **Output:** The system will return the contents of the shopping cart to the customer; the unit price and total price will be shown as well.

- **Edit Billing & Shipping Details:**
  - **Purpose:** This is to allow customers to edit and update their billing & shipping information.
  - **Input:** When the customer requests to checkout and he does not have credit card information stored at this point (system cannot find his payment information), the system will prompt credit card information page. The
customer will be given a choice on whether he wants the item shipped to his
stored address or to an alternative address.

- **Output**: The input payment information will be saved into the order form.

- **Checkout**:
  - **Purpose**: To allow user to buy the products added to the shopping cart.
  - **Precondition**: User must be logged in and must have atleast one item in
    shopping Cart to be able to checkout and place the order.
  - **Input**: When the customer finishes shopping, he requests to checkout by
    clicking “checkout” button on Cart.aspx page.
  - **Output**: If the payment information of this customer already exists, the
    system prompts the customer to review or input a new one. If the credit card
    is valid, the order form will be processed by the system and checkout is
    complete.

The *Administrator* actor can use the system for:

1) **Use Case 7 : System Login Requirement**
   - **Purpose**: This is implemented to enable user authentication. A valid user
     account must be used for an existing customer.
   - **Input**: The user will enter two inputs (user name and password) through the
     keyboard.
   - **Processing**: The user inputs will be validated and authenticated against the local
     server. The system will check the user name and password to see if they match
     the data stored onto the database.
   - **Output**: If the user name or password is invalid, the appropriate error message
     will be displayed and the user will be requested to re-enter user name and
     password. If the user inputs are valid, the default page will be displayed. If the
     user is classified as an administrator, he/she will be redirected to an
     administrator page wherein he/she can update the category details and view
     customer orders.
2) Use Case 8: Manage Catalog Requirements:

This is implemented to allow Administrator to do the following tasks:

- **Add New Genre/Category:**
  - **Purpose:** To create and add new genres, categories to the catalog.
  - **Precondition:** Administrator must be logged in to be able to create and add a new genre or category. Also, the genre to which the new category is to be associated should exist in catalog.
  - **Input:** Administrator will enter the name and necessary details to create a new genre or category to the Catalog and click “Add” button to complete the action.
  - **Output:** After the action, the changes to the catalog will be updated and saved and a message will be displayed accordingly.

- **Delete genre/category:**
  - **Purpose:** To remove genres, categories from the catalog.
  - **Precondition:** Administrator must be logged in to be able to delete a genre or category. There has to be at least one genre already present in catalog.
  - **Input:** Administrator will select a genre/category that is to be removed from the catalog and click “Remove” button.
  - **Output:** After the action, the changes to the catalog will be updated and saved and a message will be displayed accordingly.

- **Add New Product:**
  - **Purpose:** To create and add new products to the catalog.
  - **Precondition:** Administrator must be logged in to be able to create a new product. Also, the genre and/or category to which the new product is to be associated should exist in catalog.
  - **Input:** Administrator will enter the name and necessary details to create a new product to the Catalog and click “Add” button to complete the action.
- **Output**: After the action, the changes to the catalog will be updated and saved and a message will be displayed accordingly.

**Delete Product**:
- **Purpose**: To remove product from the catalog.
- **Precondition**: Administrator must be logged in to be able to delete a product. There has to be atleast one product already present in catalog.
- **Input**: Administrator will select a genre/category that is to be removed from the catalog and click “Remove” button.
- **Output**: After the action, the changes to the catalog will be updated and saved and a message will be displayed accordingly.

**Manage Orders**:
- **Purpose**: To allow the site administrator to review and manage pending and past orders according to various criteria such as date and status.
- **Precondition**: Administrator must be logged into the system. There has to be at least one order already present in database.
- **Input**: Administrator will enter the number of recent records he wishes to view and the range of dates the records are created. He/she will press the Go button against one or both the options – to view unverified, uncanceled orders and/or to view verified, uncompleted orders.
- **Output**: If the administrator enters invalid dates (Start date should be more recent then the End date) to view orders between the range, the system should display appropriate error message. The orders will be displayed as a dataset. Also, after all the orders are displayed and the administrator presses select button for an order, he/she will be redirected to OrdersAdmin Web form where he can view and update order information. When selecting an order, its details are displayed.

**Manage Shopping Carts**: 
- **Purpose:** To enable the administrator to see how many old shopping cart entries exist (shopping carts that haven’t been updated by their respective user for certain amount of time) and delete them if necessary.
- **Precondition:** Administrator must be logged into the system. There has to be at least one shopping cart before and up to that date.
- **Input:** Administrator will select the number of days from the drop-down list (e.g. 10, 15, 20 etc.) and click on “Count Old Shopping Carts” and/or “Delete Old Shopping Carts” button(s) on ShoppingCartAdmin.aspx Web form.
- **Output:** After the action, the changes to the catalog will be updated and saved and a message will be displayed accordingly.

### 3.4. Performance Requirements

The application will be used for client side caching for user inputs to reduce the load of the Web server. Also, JavaScript will be used to implement the client side validation to alleviate the task load on the server side. The system load will mostly be consisted of querying the data from the database. The Music CD data file should be about 6MB in size. The maximum response time and data transfer time between server and client will be about two minutes with a typical 56kbps per transaction. There would be approximately 100 total users. To allow maximum performance and to accommodate all user connections, the system will be built with the Microsoft SQL server instead of the test server, MSDE, which only accepts eight concurrent user connections. All users are encouraged to access the system with a broadband connection for faster performance.

### 3.5. Assumptions

- The user will have Internet connection whenever he/she is using the Online Music CD Store Website.
- The user will use Internet Explorer whenever he/she is using the Online Music CD Store Website.
- The database will be implemented using the Microsoft SQL Server.
3.6. Constraints

- The Web application is developed under the .NET Framework; therefore, the system can only be deployed on the IIS server.
- MSDE, which can only accommodate eight user connections, will be replaced by the Microsoft SQL server 2005 to allow maximum user connection.

3.7. Environment

- The Online Music CD Store Website will be written in C# language.
- The development environment will be Microsoft Visual Studio 2005.
- The Online Music CD Store Website will be tested on Windows XP platform.

4. The Case Based Reasoning Concept: Using Old Solutions for New Problems

Old problems and their solutions are stored in a database of cases – the case base. When Case Based Reasoning (CBR) is applied to recommender systems, the importance of the recommendation process lays with the case base representation - a list of experiences (cases) of the user in certain items. Experiences are represented by means of objective attributes describing the item (case definition) and subjective attributes describing implicit or explicit interests of the user in this item (case solution).

When a new problem has to be solved, the CBR system searches for the most similar old problem within the case base. The solution to this old problem can be adapted to more precisely meet the requirements of the new problem. Figure below illustrates the steps taken in a Case-Based Reasoning system.

![General Processing within a CBR System](image-url)
For applying CBR to product search in ecommerce, the cases are descriptions of products. The problem description in a case is a specification of a single product and possible demands the product can satisfy. The solution to the new problem is an unambiguous reference to the product. When a user enters a query, i.e. a description of a desired product, the query is regarded as a new problem and the CBR system tries to solve it by comparing it to cases in the case base.

However, when CBR is applied to recommender systems, the following things should be taken into consideration:

1) Firstly, case-base a vast store of old cases on which to base the decision of finding and adapting solution to the new problem. When a new problem comes up, system looks for similar problems in case base and try to solve it based on the most similar cases. However, the time dimension is also present in this reasoning process. It means that system keep in mind the most recent cases and give them the greater importance when making a decision. When dealing with human interests and preferences, the relevance of the most recent cases becomes even more important. Human interests evolve as time passes and what humans like in the present is more important than what humans liked in the past. In CBR, all the cases in the case base have the same relevance when they are retrieved. Therefore, if CBR is to be based on human interests, the relative relevance of cases according to time should be taken into account.

2) In CBR, with a larger set of cases, the system gives better results if the cases cover a wide range of problems. However, several authors claim that when the case base reaches a critical number of cases, the performance of the system does not improve but often gets worse [Leake 98]. Thus, one of the main drawbacks of CBR is the utility problem: the uncontrolled growth of case bases may degrade system performance.

4.1. Case-Based Recommendation Framework:

The core of CBR is a case base which includes all the previous experiences that can give the information which used to deal with new problems. Then, the most similar experiences or cases are retrieved. For instance, two products with the same
price would get maximum similarity if the user was interested in products with that same price, but would get very different similarity for other concepts, such as quality or trademark.

The case base represents the user profile and consists of a set of previous experiences (cases); i.e. items explicitly and/or implicitly assessed by the user (refer section 3.1). Each case contains the item description i.e. attributes describing an item and the interest attributes describing the interests of the user concerning the item. These latter attributes can be explicitly given by the user or implicitly captured by the system.

Assuming that the user’s interest in a new item is similar to the user’s interest in similar old items, in order to evaluate whether a new item could interest the user, the recommender system searches the case base for similar items. If the interest the user showed in them is high enough, the new item is recommended to the user.

4.1.1. The CBR Cycle:

The CBR cycle may be described by the following four processes:

1. RETRIEVE the most similar case or cases
2. REUSE the information and knowledge in that case to solve the problem
3. REVISE the proposed solution
4. RETAIN the parts of this experience likely to be useful for future problem solving
A new problem is solved by retrieving one or more previously experienced cases, reusing the case in one way or another, revising the solution based on reusing a previous case, and retaining the new experience by incorporating it into the existing case-base. Fig.4 illustrates the CBR process. An initial description of a problem (top of Fig.4) defines a new case. This new case is used to RETRIEVE a case from the collection of previous cases. The retrieved case is combined with the new case - through REUSE - into a solved case, i.e. a proposed solution to the initial problem. Through the REVISE process this solution is tested for success, e.g. by being applied to the real world environment, and repaired if failed. During RETAIN, useful experience is
retained for future reuse, and the case base is updated by a new learned case, or by modification of some existing cases.

4.1.2. In the following sections the different CBR phases are detailed.

4.1.2.1. The RETRIEVAL Phase

Retrieving a case means to start with a new case, and retrieve the best matching previous case. Its subtasks are to identify relevant features, find an initial set of matches, and select the most promising one.

- Initially match - The task of finding a good match is typically split into two subtasks: An initial matching process which retrieves a set of possible matches, and a more elaborate process of selecting the best one among these. Finding a set of matching cases is done by using the input problem features as indexes to the case base. There are three ways of retrieving a case or a set of cases: By following direct index pointers from problem features, by searching an index structure, or by searching in a model of previous cases.

Cases may be retrieved solely from input problem features, or also from features inferred from the input. Cases that match all input features are good candidates for matching, but cases that match only a fraction of the problem features may also be retrieved.

- Select - From the set of similar cases, a best match is chosen. This is done during the initial match process, but more often a set of cases are returned from that task. The user tries to explain away non-identical features. If the match is not good enough, a better one is sought by using links to closely related cases. The selection process can generate matches and expectations from each retrieved case, by using the case base.

4.1.2.2. The REUSE Phase

The reuse of the retrieved case solution in the context of the new case focuses on two aspects: (a) the differences among the old & the current case
and (b) what part of a retrieved case can be used in the new case. It either involves copying the old solution or adapting it:

- **Copy** - In simple classification tasks, the differences between the old and case are considered non relevant and considering only the relevant similarities, the solution is simply copied from the retrieved case to the new case. But this is a trivial type of reuse situation. However, other systems have to take into account the differences in the reused part that cannot be directly transferred to the new case but requires an *adaptation* process that takes into account those differences.

- **Adapt** - There are two main ways to reuse past cases:
  1. Reuse the past case solution (transformed) but is not directly a solution for the new case. This type of reuse does not look how a problem is solved but focuses on the equivalence of solutions.
  2. Reuse the past method that produced the solution. The retrieved case holds information about the method used for solving the retrieved problem. This type of reuse looks at how the problem was solved in the retrieved case and then re-instantiates the retrieved method to the new case.

### 4.1.2.3. The REVISION Phase

When a case solution generated by the reuse phase is not correct, the failures are to be corrected. This phase is called case revision and consists of two tasks: (1) evaluate the case solution generated by reuse phase. If successful, the system can retain the case, (2) otherwise repair the case solution using the case base.

- **Evaluate Solution** - This is usually a step outside the CBR system. The evaluation task involves the application of a proposed solution to the real environment problem. The results from applying the solution may take some time to appear, depending on the type of application. Here, the case may still be saved, and be available in the case base in the intermediate period, but it has to be marked as a non-evaluated case. A solution may
also be applied to an environment that is able to generate a correct solution. In CBR systems the solution is successful or wrong. When the solution is successful, the system retains the case, inserting it into the case base. But when the solution fails, the system is also interested in retaining the reason for the failure.

- **Repair Case Solution** - Case repair involves detecting the errors of the current solution and retrieving or generating explanations for them. This task then uses the failure explanations to modify the solution in such a way that failures do not occur.

4.1.2.4. The RETAIN Phase

This is the process of incorporating what is useful to retain from the new problem solving episode into the existing case base. It involves selecting which information from the case to retain, in what form to retain it, how to index the case for later retrieval from similar problems, and how to integrate the new case in the case base.

**Example:**

This example is to present a CBR approach that takes a song as the input and returns a recommended playlist as the output. A good playlist is not just a set of songs, but a collection of songs arranged in a sequence and these songs are similar to the input song. The CBR approach focuses on recommending similar, new and meaningful playlists, i.e. selecting a collection of songs that are arranged in a sequence and that closely match with the input song. Here, the Case Base is formed by a collection of playlists, previously experienced by human listeners. The CBR system first retrieves from the Case Base the most similar songs, their respective playlists, then combines them to generate a new playlist, both relevant to the input song and ordered.

Case-Based Reasoning approach comprises four subsequent steps:

(1) Assemble a Case Base from a repository of playlists (Case Base Setup).
(2) Prompt the user for an input song $s$ and the desired length $l$ of the similar song list (Problem Description).

(3) Retrieve from the Case Base a subset of songs with length $l$ that satisfy most of the goals. Specifically, the retrieved songs are most closely matched to the input song (Retrieve Process).

(4) Combine the playlists of the retrieved similar songs to generate a ordered recommendation list $P$ (Reuse Process).

### Fig. General View of the CBR Process

**Step1) The Case-Base Setup**

Every playlist of Case-Base is a sequence of songs: $P = (s_1, s_2, \ldots, s_n)$. Every song is characterized by a set of attributes e.g.: an identifier of the track, an identifier of the album it belongs to, the artist that performs it, etc. Here it is assumed that the recommendations where close songs repeat the same attribute: for example playlists where many songs belong to the same album or artist are approved for playlists.

**Case Base Setup example:**

A sample collection of 5 playlists and their recommended songs by different users
Step 2) RETRIEVE Process:
Retrieve process is focused on finding which songs in the Case Base are the most similar to the input song from the previous users’ playlists that are useful to achieve the requested goals. The idea is to retrieve a subset of those playlists that include $s$, are varied, and are sequentially ordered.

E.g. (i)

Given the input song name as *Numb*, artist as *Mat Kearney*, style *Pop* and year of release as *2005*, which are the best 10 most similar songs to retrieve out of the available songs in database?

(iii)
(iii) The Most similar songs selected are:

- Everything goes numb
  Scissor Sisters
  Rap
  2005

- Stand
  R.E.M
  Vocal
  2004

- Creep
  Radiohead
  Classical
  2004

- Numb America
  Chaser
  Rock
  2004

- Dumb Numb
  Friction
  Pop
  2004

- Trouble
  Harnakis
  Rock
  2003

- Numb to this
  Gary Ward
  Jazz
  2006

- Numb Nuts
  Snuff
  Vocal
  2006

- Numb Right
  The Beatles
  Classical
  2004

- Travis
  String
  Rap
  2005

- Warm Strong Numb
  Lemon
  Rock
  2003

- Raising Sand
  Robert Plant
  Folk
  2006

- Carnival Ride
  Carrie
  Jazz
  2004

- Search for Numb
  Friction
  Rap
  2004

- One Chance
  Paul Potts
  Folk
  2005

- Wonderwall
  Oasis
  Pop
  2007

- Living Numb
  Soulfix
  Rock
  2006

- Let it snow
  Michael Buble
  Jazz
  2007

- Lament for the Numb
  James Taylor
  Vocal
  2004

- Numb ’s the word
  Rick Boston
  Pop
  2005

- Comfortably Numb
  Scissor Sisters
  Rap
  2005

- Numb Nuts
  Snuff
  Vocal
  2006

- Numb America
  Chaser
  Rock
  2004

- Dumb Numb
  Friction
  Pop
  2004

- Warm Strong Numb
  Lemon
  Rock
  2003

- Numb to this
  Gary Ward
  Jazz
  2006

- Living Numb
  Soulfix
  Rock
  2006

- Search for Numb
  Friction
  Rap
  2004

- Numb ’s the word
  Rick Boston
  Pop
  2005

- Let it snow
  Michael Buble
  Jazz
  2007

- Lament for the Numb
  James Taylor
  Vocal
  2004

- Numb ’s the word
  Rick Boston
  Pop
  2005
(iv) From the above songs, the user selects following songs for which he wants to see the recommendations:

<table>
<thead>
<tr>
<th>Warm Strong Numb</th>
<th>Lemon</th>
<th>Rock</th>
<th>2003</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numb to this</td>
<td>Gary Ward</td>
<td>Jazz</td>
<td>2006</td>
</tr>
<tr>
<td>Living Numb</td>
<td>Soulfix</td>
<td>Rock</td>
<td>2006</td>
</tr>
<tr>
<td>Search for Numb</td>
<td>Friction</td>
<td>Rap</td>
<td>2004</td>
</tr>
<tr>
<td>Numb’s the word</td>
<td>Rick Boston</td>
<td>Pop</td>
<td>2005</td>
</tr>
</tbody>
</table>

| Everything goes numb | Scissor Sisters | Rap | 2005 |
| Comfortably Numb     | Scissor Sisters | Rap | 2005 |
| Numb Nuts            | Snuff          | Vocal | 2006 |
| Numb America         | Chaser         | Rock | 2004 |
| Dumb Numb            | Friction       | Pop | 2004 |

**Step 3) REUSE Process:**

The goal of the Reuse process is to transform the retrieved playlists into a single, recommended playlist of a specific length $l$. But the retrieved playlists can have any length, so they cannot be directly returned as recommendations. For Reuse, the songs from the retrieved playlists are used to build a new combination of playlist which will be the desired recommendation playlist to the input song.
E.g:
(i) The playlists of only the selected songs are used to derive the recommendation list.

All the songs in each the above playlists are matched with the input song, arranged in ascending order according to the style-distance + artist-distance and top 5 songs are selected as recommendations to the user.

Then it is returned as a recommended playlist

**For the above process following assumptions are made:**

1) During the retrieval phase, when the input song is matched with the songs in the database, only the key words of the song name are matched and the words (stop words) like ‘and’, ‘or’, ‘the’, ‘but’, ‘is’, ‘of’ etc are omitted. In this case, for the input song ‘Anything but love’, the word ‘but’ would be omitted and only ‘Anything’ & ‘love’ will be matched with the track names from the database to get
the list of songs whose names contain these words. The attributes of these songs will then be matched with attributes of the input song (artist, style and year).

2) In the retrieval phase, if a song is selected from the database by matching the input song, all the songs belonging to its album will be selected irrespective of the matches in the song name or their attributes. But if there are too many songs in the same album, only those songs whose distance is minimum (style-distance and artist distance) with the input song will be selected.

3) The following are the assumed distances between styles:

\[
\begin{align*}
(\text{style1, style2} = \text{distance}) & \implies \\
(\text{Jazz, Classical}) & = 5 \\
(\text{Jazz, Country}) & = 4 \\
(\text{Jazz, Folk}) & = 6 \\
(\text{Jazz, Jazz}) & = 0 \\
(\text{Jazz, Karaoke}) & = 3 \\
(\text{Jazz, Latin}) & = 5 \\
(\text{Jazz, Pop}) & = 1 \\
(\text{Jazz, Rap}) & = 2 \\
(\text{Jazz, Rock}) & = 1 \\
(\text{Jazz, Techno}) & = 3 \\
(\text{Jazz, Vocal}) & = 2 \\
(\text{Karaoke, Techno}) & = 1 \\
(\text{Latin, Classical}) & = 3 \\
(\text{Latin, Pop}) & = 1 \\
(\text{Rap, Latin}) & = 2 \\
(\text{Rap, Techno}) & = 1 \\
(\text{Techno, Pop}) & = 1
\end{align*}
\]

4) The following are the assumed distances between artist of the input song and other:

\[
\begin{align*}
(\text{artist1, artist2} = \text{distance}) & \implies \\
(\text{Barry White, Diana Krall}) & = 2 \\
(\text{Barry White, Dr.Evil}) & = 4 \\
(\text{Barry White, Torley Wong}) & = 6 \\
(\text{Barry White, Vincent Music}) & = 4 \\
(\text{Barry White, Barry White}) & = 0 \\
(\text{Barry White, Giant Sand}) & = 7 \\
(\text{Barry White, Eric Serra}) & = 1 \\
(\text{Barry White, Willie Nelson}) & = 1 \\
(\text{Barry White, Bob Dylan}) & = 4 \\
(\text{Barry White, Tom Waits}) & = 1
\end{align*}
\]
5) The matched song would lie within a range of $\pm2$ years of that of the song with which it is being matched. For e.g. if the song to be matched has year of release 2005, the songs that have year of release from 2003 to 2007 (i.e. $\pm2$ years of 2005) only will be selected.

6) Each song in the Track table will have 5 recommended songs which will form the case-base of the process.

7) The user will be displayed a list of 10 most similar songs (to that of the input song) from which he can select the songs of his choice to get recommendations.

8) The user will be displayed 5 most recommended songs to the songs he has selected from the retrieved similar songs.
Chapter 2 – Project Plan

1. Task Breakdown

1.1. Inception Phase

The inception phase is focused on defining the project requirements. The primary documents to be created in this phase include the Vision document, SQA Plan and the project plan.

- *Vision document* will include an overview of the project, its purpose, goals, risks, constraints, and direction. It will also discuss the main product features, quality attributes, and external interfaces. It will also include the critical project requirements and the major use cases will be defined and elaborated in the requirements analysis.

- *The Project plan* developed will describe the work to be accomplished in each phase as well as the inclusion of an estimate of the workload of the project that will establish a schedule for the completion of all project activities.

- *The Software Quality Assurance (SQA) Plan* will describe the required documentation, standards and conventions, test tracking and problem reporting, and tools used during the project. The plan will also identify the set of quality metrics used to assess product reliability. A simple prototype will be built during this phase so as to establish the project feasibility.

An executable prototype of the user interface will be demonstrated in Presentation I to establish the feasibility of the important elements of the use case requirements. This will be a milestone for the inception phase.

1.2. Elaboration Phase

The elaboration phase concentrate on the architecture design of the system. The complete architectural design will be documented using appropriate UML diagrams. Each component in the architecture will be documented at the interface level. Reuse of commercial or pre-existing components will be documented. Revisions will be made to the initial vision document to provide a complete representation of all requirements, and the project plan based on the feedback from
the committee members. A project component will be formally specified using a published, formal methodology. A test plan will be developed to outline all testing activities and how to report and track those test results. The two technical inspectors will perform an architecture review and provide feedback by submitting a formal report based on their findings.

As a conclusion of this phase, the developer will demonstrate another executable prototype to illustrate more product features and functionality and submit the required documentation for approval by the supervisory committee.

1.3. Production Phase

The production phase is concentrated on the implementation design requirements, deployment and testing of the system. In this phase, the developer will construct the code and ensure that it is well documented. The code will be tested entirely to guarantee that all requirements are met. All test results will be analyzed and documented. A user manual will also be produced by the developer, which will describe how to install, run, and use the tool efficiently.

At the conclusion of this phase, the developer will present the final version of the software product as the final presentation as well as submit all the required documentation. Review and approval of final presentation determines the completion of project.

The table below lists all the adjustment factors and their corresponding ranges.

<table>
<thead>
<tr>
<th>IDENTIFIER</th>
<th>EFFORT ADJUSTMENT FACTOR</th>
<th>Very Low</th>
<th>Low</th>
<th>Nominal</th>
<th>High</th>
<th>Very High</th>
<th>Extra High</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELY</td>
<td>Required software reliability</td>
<td>0.75</td>
<td>0.88</td>
<td>1.00</td>
<td>1.15</td>
<td>1.40</td>
<td></td>
</tr>
<tr>
<td>DATA</td>
<td>Size of application database</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.94</td>
<td>1.00</td>
<td>1.08</td>
<td>1.16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPLX</td>
<td>Complexity of the product</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.70</td>
<td>0.85</td>
<td>1.00</td>
<td>1.15</td>
<td>1.30</td>
<td>1.65</td>
</tr>
<tr>
<td>TIME</td>
<td>Runtime performance constraint</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.00</td>
<td>1.11</td>
<td>1.30</td>
<td>1.66</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STOR</td>
<td>Memory constraints</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.00</td>
<td>1.06</td>
<td>1.21</td>
<td>1.56</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Description</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>----------------------------------</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VIIRT</td>
<td>Virtual machine volatility</td>
<td>0.87</td>
<td>1.00</td>
<td>1.15</td>
<td>1.30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TURN</td>
<td>Required turnabout time</td>
<td>0.87</td>
<td>1.00</td>
<td>1.07</td>
<td>1.15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACAP</td>
<td>Analyst capability</td>
<td>1.46</td>
<td>1.19</td>
<td>1.00</td>
<td>0.86</td>
<td>0.71</td>
<td></td>
</tr>
<tr>
<td>AEXP</td>
<td>Applications experience</td>
<td>1.29</td>
<td>1.13</td>
<td>1.00</td>
<td>0.91</td>
<td>0.82</td>
<td></td>
</tr>
<tr>
<td>PCAP</td>
<td>Software engineer capability</td>
<td>1.42</td>
<td>1.17</td>
<td>1.00</td>
<td>0.86</td>
<td>0.70</td>
<td></td>
</tr>
<tr>
<td>VEXP</td>
<td>Virtual machine experience</td>
<td>1.21</td>
<td>1.10</td>
<td>1.00</td>
<td>0.90</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LEXP</td>
<td>Language experience</td>
<td>1.14</td>
<td>1.07</td>
<td>1.00</td>
<td>0.95</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOOL</td>
<td>Use of software tools</td>
<td>1.24</td>
<td>1.10</td>
<td>1.00</td>
<td>0.91</td>
<td>0.82</td>
<td></td>
</tr>
<tr>
<td>MODP</td>
<td>Use of Modern Practices</td>
<td>1.24</td>
<td>1.10</td>
<td>1.00</td>
<td>0.91</td>
<td>0.83</td>
<td></td>
</tr>
<tr>
<td>SCED</td>
<td>Required development schedule</td>
<td>1.23</td>
<td>1.08</td>
<td>1.00</td>
<td>1.04</td>
<td>1.10</td>
<td></td>
</tr>
</tbody>
</table>

Adjustment factors for the Online Music CD Store are as follows:

- RELY as nominal and a value of 1.00
- DATA as high and a value of 1.08
- CPLX as low and a value of 0.85
- TIME as nominal and a value of 1.00
- STOR as low and a value of 1.00
- VIRT as nominal and a value of 1.05
- TURN as low and a value of 0.87
- ACAP as high and a value of 0.8
- AEXP as nominal and a value of 1.00
- PCAP as nominal and a value of 1.00
- VEXP as nominal and a value of 1.00
- LEXP as nominal and a value of 1.00
- MODP as high and a value of 0.91
- TOOL as high and a value of 0.91
- SCED as nominal and a value of 1.00

45
The EAF value is calculated to 0.61. I estimated the size to be around 2500 LOC based on the current prototype and similar examples.

The effort evaluates to:

\[
\text{Effort} = 3.2 \times 0.61 \times (2.5)^{1.05} = 5.11 \text{ staff months}
\]

The time can now be calculated as:

\[
\text{Time} = 2.5 \times (5.11)^{0.38} = 4.64 \text{ months (development time)}
\]

2. **Architecture Elaboration Plan**

   The following tasks have to be completed in the elaboration phase, before the second presentation is given.

2.1. **Revision of Vision Document**

   The Vision Document will be revised to provide a complete representation of requirements. These requirements will be ranked according to importance, and a set of “critical” requirements identified. The document revision will be based from the feedback given by the committee members after the first presentation. The corrected version of the document will be submitted to the major professor for approval.

2.2. **Revision of the Project Plan**

   The Project Plan will be revised based on the feedback provided by the committee members after the first presentation. The document will provide an updated estimate on the size, cost and effort required for the project implementation. It will also contain the Implementation plan which will define the activities and actions that must be accomplished during implementation. The plan will include a Work Breakdown Structure, complete with time and costs estimates and completion criteria. The updated version will be submitted to the major professor for approval.

2.3. **Formal Requirements Specification**
The Object Constraint Language (OCL) and UML-based Specification Environment (USE) will be used to define and verify the formal specification of the product. At least one component of the design will be formally specified using the methodologies mentioned above. As of now, it has been proposed that a part of the object model will be formalized.

2.4. **Architecture Design**

The complete architectural design will be documented using appropriate diagrams such as class and object diagrams, sequence/collaboration diagrams, statechart/activity diagrams, hierarchy diagrams, etc. Each component in the architecture will be documented at the interface level.

2.5. **Test Plan**

A plan will be developed for the project to address the required tests to show that the product satisfies the requirements. The plan will include evaluation criteria for all critical use cases and a set of test data deemed adequate for acceptance testing. Specifically, the test plan will identify a set of test cases, the types of tests that will be used for these test cases, the data that will be used for each case, and the requirement traces for each test case.

2.6. **Formal Technical Inspection**

The above artifacts will be subjected to a formal technical inspection by two independent MSE students, Rahul Deshmukh and Srunokshi Neelkantan. A formal checklist to be used by the inspectors will be prepared. Each independent inspector will provide a report on the result of their inspection and these reports will become part of the project documentation.

2.7. **Architecture Prototype**

Prior to the Presentation II, an executable architecture prototype will be built that will address all critical requirements identified in the vision document.
3. Cost Estimate

3.1. COCOMO Model

In 1981, Barry Boehm designed "Constructive COst MOdel" to give an estimate of the number of person-months it will take to develop a software product. The model also estimates the development schedule in months and produces an effort and schedule distribution by major phases. The model estimates cost using one of three different development modes: organic, semidetached and embedded. Organic projects - are relatively small, simple software projects in which small teams with good application experience work to a set of less than rigid requirements. Semi-detached projects - are intermediate (in size and complexity) software projects in which teams with mixed experience levels must meet a mix of rigid and less than rigid requirements. Embedded projects - are software projects that must be developed within a set of tight hardware, software, and operational constraints.

The Online Music CD Store will be a small scale application of average complexity and fair flexibility. Therefore, it is classified as an organic mode project under the COCOMO model. The basic COCOMO equations for organic projects take the form:

\[
\text{Effort} = 3.2 \times \text{EAF} \times (\text{Size})^{1.05}
\]

\[
\text{Time} = 2.5 \times (\text{Effort})^{0.38}
\]

Where:

- **Effort** = number of staff months (PM)
- **EAF** = effort adjustment factor
- **Size** = number of lines of code for completed product. It is measured in KLOC (thousands of lines of codes)

\[
\text{Time} = \text{total number of months}
\]

The Effort Adjustment Factor is the product of the 15 adjustment parameters. Each of the 15 attributes receives a rating on a 6-point scale that ranges from "very low" to "extra high" (in importance or value). An effort multiplier from the table below applies to the rating. The product of all effort multipliers results in an 'effort adjustment factor (EAF). Typical values for EAF range from 0.9 to 1.4.
3.2. GANTT Chart

![GANTT Chart](image)

Fig5. The Project Gantt Chart

4. Implementation Plan

4.1. Deliverables

The following are the deliverables for Presentation III:

**Source Code**

Well-documented source code will be submitted. This code will correspond directly to the architecture and component design.

**Assessment Evaluation**

Test cases that were identified in the Test Plan will be executed. Defect fixing will be done for all resolvable issues. All test results will be documented. The
documentation will include a document detailing the testing done on the project along with the descriptions of the testing and known unresolved defects.

**User Manual**

The user manual will include an installation guide and a user guide. The installation guide will include detailed information on how to set up the software, while the user guide will give detailed explanations of common usage and user commands.

**Component Design**

The internal design of each component will be documented. UML diagrams will be used for the said design.

**Project Evaluation**

The entire software process starting from the first phase will be reviewed. This will include the usefulness of the methodologies used, the accuracy of the estimations, and the usefulness of the reviews. The tool will be reviewed and evaluated to check whether it accomplishes the goals presented in the vision document and if the quality of the product is achieved.

**Formal Technical Inspection Letters**

The two MSE students will perform the technical inspection of the architecture design. They will submit a formal letter stating that the project has successfully passed all technical requirements.

**References**

The annotated bibliography will include cited references for all notations used in the portfolio.

**4.2. Work Breakdown Structure**

The following table breaks down the deliverables into tasks and lists the completion criteria and cost for each task.
<table>
<thead>
<tr>
<th>Deliverable</th>
<th>Tasks</th>
<th>Completion Criterion</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Code</td>
<td>Develop Customer’s functions</td>
<td>Executable code</td>
<td>15 days</td>
</tr>
<tr>
<td></td>
<td>Develop Administrators’s functions</td>
<td>Executable code</td>
<td>11 days</td>
</tr>
<tr>
<td>Assessment Evaluation and Testing</td>
<td>Run test cases and fix resolvable issues</td>
<td>All test cases complete</td>
<td>4 days</td>
</tr>
<tr>
<td></td>
<td>Document test results</td>
<td>All the test case results evaluated and documented</td>
<td>2 days</td>
</tr>
<tr>
<td>User Manual</td>
<td>Installation Guide</td>
<td>Approved by Major Professor</td>
<td>1 day</td>
</tr>
<tr>
<td></td>
<td>User Guide</td>
<td>Approved by Major Professor</td>
<td>1 day</td>
</tr>
<tr>
<td>Component Design</td>
<td>Document Online Music CD Store Design</td>
<td>All major features of the Online Music CD Store are documented</td>
<td>2 days</td>
</tr>
<tr>
<td></td>
<td>Document usefulness of project, methodologies, practices, and reviews</td>
<td>Approved by Major Professor</td>
<td>2 days</td>
</tr>
<tr>
<td></td>
<td>Compile all project resources together</td>
<td>Approved by Major Professor</td>
<td>2 days</td>
</tr>
<tr>
<td>Project Evaluation</td>
<td>Receive letters from formal technical inspectors</td>
<td>Approved by Major Professor</td>
<td>1 day</td>
</tr>
<tr>
<td>References</td>
<td>All references documented</td>
<td>Approved by Major Professor</td>
<td>1 day</td>
</tr>
</tbody>
</table>

Table 1 The Work Breakdown Structure Table
Chapter 3 – Architecture Design

1. Introduction

The purpose of this document is to provide an architectural design for the Online Music CD Store. The design will show the presentation tier, the middle tier, composing of class and sequence diagrams, and the data tier. Each class will have a brief description about its purpose.

2. Architecture of the system

The architecture of the system is based on three-tier architecture. There are three logical tiers: the Presentation Tier, the Business Tier, and the Data Tier. The presentation tier is responsible for displaying the contents to the user while business tier communicates between presentation tier and data tier, for example receiving the search request for the product and passing it to the database and submitting the results back to presentation tier for display, or displaying the cart empty message when there is no product in the cart. It is the main component of the website since it is responsible for handling the overall logic. The data tier is responsible for storing the product information, transactions of the customer and shopping cart products. The data tier is also responsible handling the requests from the business tier and using stored procedures passing the result back.

The figure below shows the three tier-architecture of the Online Music CD Store.
2.1. Presentation Tier

The presentation tier for the system is ASP.NET Web forms with User Controls. The Visual Studio .NET IDE will be used to create the Web forms. It uses code behind code, where the code for each ASPX page is encapsulated into a separate file.

The table below shows ASP.NET Web forms for clients of the system.

<table>
<thead>
<tr>
<th>ASP.NET Web Forms</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default.aspx</td>
<td>The Web page for welcome screen.</td>
</tr>
<tr>
<td>Register.aspx</td>
<td>The Web page where a new user can create a user account.</td>
</tr>
</tbody>
</table>
Login.aspx  The Web page for login to secure Web sites

CustomerDetails.aspx  The Web page where the users edit their profile information.

Catalog.aspx  The Web page displays different genres, their respective categories and the collection of products (Music CDs available).

Product.aspx  The Web page displays information of the product selected by the user along with their recommendations.

Search.aspx  The Web page for searching an available product.

ShoppingCart.aspx  The Web page displays the details of the shopping cart to which customer have added the products to buy.

Checkout.aspx  The Web page where user can place an order.

OrderPlaced.aspx  The Web page for confirming a purchase order.

<table>
<thead>
<tr>
<th>ASP.NET Web Forms</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Login.aspx</td>
<td>The Web page for login to secure Web sites</td>
</tr>
<tr>
<td>Index.aspx</td>
<td>The Web page for welcome screen.</td>
</tr>
<tr>
<td>CatalogAdmin.aspx</td>
<td>The Web page for adding new or updating the existing quantity and details of genres/categories/products.</td>
</tr>
<tr>
<td>OrdersAdmin.aspx</td>
<td>The Web page for viewing order history and managing the customer order details.</td>
</tr>
<tr>
<td>ShoppingCartAdmin.aspx</td>
<td>The Web page for managing (deleting the very old ones) the customer shopping carts.</td>
</tr>
</tbody>
</table>

Table 2 Client Site ASP.NET Web Forms

The table below shows ASP.NET Web forms for the administrator of the system.

Table 3 Administrator Site ASP.NET Web Forms
2.2. Business Tier

The business specific layer consists of fifteen classes: User, Customer, Administrator, SessionManager, Genre, Product, ShoppingCart, CartItem, Order, OrderDetail, ShippingInfo, searchFacade, keyword, productSet, RecommendationSet.

The diagram below captures the domain model of the Online Music CD Store.

Fig7. Domain Model

The diagram below captures the page flow (ASP.NET Web forms) model for the user.
The diagram below captures the page flow (ASP.NET Web forms) model for the Administrator.

2.2.1. Class Descriptions

2.2.1.1. User

<table>
<thead>
<tr>
<th>User</th>
</tr>
</thead>
<tbody>
<tr>
<td>• userId : string</td>
</tr>
<tr>
<td>• password : string</td>
</tr>
<tr>
<td>• loginStatus : string</td>
</tr>
<tr>
<td>• verifyLogin()</td>
</tr>
</tbody>
</table>
This class will handle all user actions. The User class is the super class of Customer and Administrator. It includes a private method to verify the login. The verifyLogin method is called when the user presses the sign in button on the Login.aspx Web form. It returns true if the login is successful, false if it is not.

### 2.2.1.2. Customer

![Customer Class Diagram](image)

This class will handle the customer actions. It inherits all the User class responsibilities and its functions. It includes the private methods for customer to register, login, update his/her personal information and search through the catalog. The login and register methods will be called when the user presses the login and register button respectively on the Default.aspx Web form. The updateProfile method is called when user clicks Edit button on CustomerDetails.aspx web form to change, update and save his/her personal information for future use. Also, the search method is called when user searches for a particular product on Search.aspx Web form.

### 2.2.1.3. Administrator
The SessionManager class supports the User’s required operations like getUser() and getAllGenres(). The getUser method is called when the user information needs to be displayed. The getAllGenres() method is used to get the existing genres in the catalog. It returns the genre(s) as a dataset. When the user needs to get a product, the system has to start from the Session Manager and navigate through Genre class down to the Product class.

2.2.1.4. SessionManager

The SessionManager class supports the User’s required operations like getUser() and getAllGenres(). The getUser method is called when the user information needs to be displayed. The getAllGenres() method is used to get the existing genres in the catalog. It returns the genre(s) as a dataset. When the user needs to get a product, the system has to start from the Session Manager and navigate through Genre class down to the Product class.

2.2.1.5. Genre
This class represents a Genre. It has private method to display products from a particular genre. The getAllProducts method is called when the user presses a genre. It returns the products from that genre as a dataset.

2.2.1.6. Product

This class represents collection of products of a particular category and/or genre. The displayProduct and getProductDetails method are called when the user clicks on a product or on the "Search" button on the Search.aspx Web form. The displayProduct method is used to retrieve image of the product and getProductDetails method retrieves its details as a dataset whenever a customer clicks on the product. A product can belong to one or more departments and/or categories.

2.2.1.7. Shopping Cart
ShoppingCart class has all the products that are added by a customer to buy. The addCartItem and deleteCartItem methods are called when customer clicks "Add to Cart" and "Delete" button on the ShoppingCart.aspx Webpage. It returns true if the addition and deletion of the products to the shopping cart is successful and false if not. The updatequantity method is called when Customer clicks "Update" button on ShoppingCart.aspx Webpage to increase or decrease the number of products in the cart. It returns true if the update operation is successful and false if not. Also, viewCartDetails method is called when customer clicks on the "View Details" button to see a summary of the cart. It will display the summary only when cart is not empty, else it will display "Your Cart is empty." The checkout method is called when the user has finished adding products to the shopping cart and presses the checkout button to buy the products. The calcUnitPrice method is called to calculate the price of each product (depending upon its quantity) that is being added to the shopping cart. The calcTotalPrice is called to calculate the total price of all products that are added to shopping Cart.

2.2.1.8. CartItem
2.2.1.9. Order

This class will store all information regarding the orders made by each customer. The placeOrder method is called when customer clicks on the "Place Order" button on the checkout.aspx Web form. It returns true if the order is placed successfully, false if it is not. The "Place Order" button will be enabled only when customer has a valid shopping cart and has entered valid personal, billing and shipping details.

2.2.1.10. OrderDetail
This Class handles details regarding every order that the user makes.

2.2.1.11. ShippingInfo

This class handles the shipping information regarding every customer and their orders. The updateShippingInfo method is called when customer edits his/her shipping information and clicks the "Update" button on CustomerDetails.aspx Web form.

2.2.1.12. keyword

This class represents a collection of all the keywords. It has a private method to match the keyword (entered by the user) with the keywords stored in the database.
2.2.1.13. searchFacade

![searchFacade Class Diagram](image)

This class has methods to get the search results according to the options chosen by the user. The getbyartist method is called when the user selects artist as his/her option as a search criteria and presses the search button on the Search.aspx Web form. The getbykeyword and getbyproductName methods are called when the user does not select any of the options from the dropdown list and clicks search button on Search.aspx Web form. It returns the product(s) as a dataset.

2.2.1.14. RecommendationSet

![RecommendationSet Class Diagram](image)

This class represents a collection of products recommended for the product selected by the user. The getrecs method is called to get a set of products that are recommended when the user clicks on a particular product. It returns the product(s) as a dataset.

2.2.1.15. ProductSet
This class represents a collection of all the products that were searched by the user as well as all the products that were recommended to the product of user’s choice. The MatchtoProduct method is called to match the details of the product with the products in database with the help of keyword class. It returns the product(s) as a dataset.

2.2.2. Sequence Diagrams

2.2.2.1. Login

The user needs to log in to the system for accessing secure sites. The user will enter his/her userid and password then clicks on the login button on the Login.aspx Web form. The SessionManager class gets the user form the User class and its password provided is verified with the User class instance it gets back. If the userid and password are correct, the system will direct the
user to appropriate Web page. Otherwise, the system will prompt the user to check the userid and password.

### 2.2.2.2. Buying (Add to Cart & Place Order)

The first part of the above sequence diagram shows a successful transaction by the user of adding a product to shopping cart. Here the user is not necessarily logged into the system to be able to browse through the site. When the user selects a product and presses ‘Add to Cart’ button on
Product.aspx Web form, the addCartItem method is called from the ShoppingCart class which then call the new() which creates a new object of the cartItem class and the product is added into shopping cart.

The second part of the above sequence diagram shows a successful transaction by a user to place an order for the products selected in shopping cart. After the user has finished adding all the selected products to the shopping cart, he presses the checkout button on the ShoppingCart.aspx Web form. Here if the user is logged in, the system redirects him/her to Login.aspx Web form. After logging in, the user can edit/update his/her personal and/or shipping information. The calcTotalPrice is then called which calculates the total price of all the products in shopping cart along with the shipping, tax charges. The “Place Order” button on Checkout.aspx Web form will then be enabled for the user to place an order.

2.2.2.3. Search

![System Search Sequence Diagram](image)

Fig.27. System Search Sequence Diagram

The sequence diagram shows a user has already logged in and has successfully searched an available item. The user will enter a keyword in the
Textbox and select options from four categories: Artist, Album, Song, and Movie. He/she will then press the search button. The findkeyword method is then called to retrieve the entered keyword from the keyword table. The MatchtoProduct method is then called to find a particular product being searched. If the result set is not null, the available product(s) will be displayed on the same Web form. Otherwise, if the result set is null, the system will prompt an appropriate error message to the user.

2.3. Data Tier (Database)

The system database has the following structure of tables:

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>Represents the user information</td>
</tr>
<tr>
<td>SessionManager</td>
<td>Contains user and catalog-genre information</td>
</tr>
<tr>
<td>Administrator</td>
<td>Represents the administrator information</td>
</tr>
<tr>
<td>Customer</td>
<td>Represents the customer information</td>
</tr>
<tr>
<td>Genre</td>
<td>Represents the genre information</td>
</tr>
<tr>
<td>Product</td>
<td>Represents the product information</td>
</tr>
<tr>
<td>ShoppingCart</td>
<td>Represents the shopping cart information</td>
</tr>
<tr>
<td>Order</td>
<td>Represents the customer order information</td>
</tr>
<tr>
<td>OrderDetail</td>
<td>Represents the order information for each order</td>
</tr>
<tr>
<td>ShippingInfo</td>
<td>Represents the shipping details for each order</td>
</tr>
<tr>
<td>RecommendationSet</td>
<td>Represents the recommended product’s information</td>
</tr>
<tr>
<td>ProductSet</td>
<td>Represents the recommended and searched product’s information</td>
</tr>
</tbody>
</table>

Table 3 Database Table

The complete physical database schema for MusicCD Store is illustrated in the database diagram below:
3. **Formal Specification for the system**

```plaintext
define model MusicCDStore

-- ENUMERATIONS

enum OrderStatus {verified, unverified, uncanceled, uncompleted}

-- CLASSES

class User
```
attributes
  userId:string
  password:string
  loginStatus:string
operations
  verifyLogin(userId:string, password:string):Boolean =
      User.allInstances->exists(u:User | u.userId = userId and u.password = password)
End

class Administrator < User
attributes
  adminId:string
  password:string
  adminEmail:string
  adminName:string
operations
  createGenre():Boolean
  createProduct():Boolean
  deleteGenre():Boolean
  deleteProduct():Boolean
  editCatalogDetails():Boolean
  viewOrderHistory():Boolean
End

class Customer < User
attributes
  customerId:string
  password:string
customerName: string
email: string
address: string
phoneno: integer
creditcardInfo: string
shippingInfo: string

operations
register(): Boolean
login(): Boolean
search()

updateProfile(customerId: string, customerName: string, email: string,
address: string, phoneno: integer): Boolean
pre: Customer.allInstances.customerId->includes(customerId)
post: Customer.allInstances.customerId = user.allInstances.customerId@pre
post: Customer.allInstances.customerName =
    user.allInstances->select(C:Customer | C.customerId <>
customerId).customerName@pre->including(customerName)
post: Customer.allInstances.email = Customer.allInstances->
select(C:Customer| C.customerId<>customerId).email@pre->including(email)
post: Customer.allInstances.address =
    Customer.allInstances->select(C:Customer |
    C.customerId<>customerId).address@pre->including(address)
post: Customer.allInstances.phoneno =
    Customer.allInstances->select(C:Customer |
    C.customerId<>customerId).phoneno@pre->including(phoneno)

End

class SessionManager
attributes
userIds: string
genreId: string
operations

getAllGenres(genreId:string):Set(Genre) = Genre.allInstances->select(g:Genre |
g.genreId = genreId)

getUser(eID:String):Set(User) = if User.allInstances->exists(u:User |
  u.userId = userId) then User.allInstances->select(u:User | u.userId = userId)
  else
    oclEmpty(Set(User))
  endif
End

class Genre
attributes
  genreId:string
  genreName:string
  description:string
operations
  getAllProducts(productId:int):Set(Product) =
    Product.allInstances->select(p:Product | p.productId = productId)
End

class Product
attributes
  productId:integer
  productName:string
  genreName:string
  artist:string
  description:string
  price:integer
  imageFileName:string
operations
getProductDetails()
End

class ShoppingCart
attributes
cartId:integer
productId:integer
quantity:integer
dateAdded:integer

Item: CartItem

operations
addCartItem(customerId : string, cartItem:CartItem)
post: Customer.allInstances -> forAll(c | c.customerId=customerId implies
c.hasCart.Item = c.hasCart.Item@pre->including(cartItem))
deleteCartItem(customerId : string, productId : integer)
post: Customer.allInstances -> forAll(c | c.customerId=customerId implies
c.hasCart.Item = c.hasCart.Item@pre->excluding(cartItem))
updatequantity(productId : integer, quantity : integer)
viewCartDetails()
checkout()
calcUnitPrice()
calcTotalPrice()
End

class CartItem
attributes
cartId:integer
name:string
class Order
attributes
    orderId:integer
    customerId:string
    dateCreated:string
    dateShipped: string
    status:OrderStatus
    shippingId:integer
operations
    new(customerId : string, productId : integer)
    post: Customer.allInstances -> forAll(c:Customer | c.customerId=customerId
        implies c.hasCart.hasCartItems.productId =
        c.hasCart.hasCartItems.productId@pre->including(productId))
    post: CartItem.allInstances.customerId =
        CartItem.allInstances.customerId@pre->including(customerId)
    post: Customer.allInstances -> forAll(c | c.customerId=customerId and
        Product.allInstances -> forAll(p| p.productId=productId implies
        c.hasCart.hasCartItem.name =
        u.hasCart.hasCartItem.name@pre->including(p.productName)))
    post: Customer.allInstances -> forAll(c | c.customerId=customerId and
        Product.allInstances -> forAll (p| p.productId=productId implies
        c.hasCart.hasCartItem.unitCost=c.hasCart.hasCartItem.unitCost
        @pre->including(p.price)))
End
placeOrder(O:Order):Boolean
pre: O.User.verifyLogin(O.User.userId, O.User.password) = true
pre: Order.allInstances->excludes(O)
post: Customer.allInstances ->
    forAll(c:Customer | c.customerId=O.customerId
        implies O.orderId = u.hasCart.orderId)
post: Order.allInstances.orderId =
    Order.allInstances.orderId@pre->including(O.orderId)
End

class ShippingInfo
attributes
    orderId:integer
    shippingId:integer
    shippingType:string
    shippingCost:integer
    shippingRegionId:integer
operations
    updateShippingInfo()
End

class OrderDetail
attributes
    orderId:integer
    productId:integer
    productName:string
    quantity:integer
    unitCost:float
    subTotal:float
End
--

-- ASSOCIATIONS
--

- each product belongs to only one Genre
  association ProductGenre between
    Genre[1] role inGenre
    Product[1..*] role hasProducts
  End

- each Shopping cart belong to only one customer
  association CustomerHasShoppingCart between
    Customer[1] role customer
    ShoppingCart[0..*] role hasCart
  End

- each shoppingcart has atleast one cartItem
  association ShoppingCartHasItem between
    ShoppingCart[1] role inCart
    CartItem[1..*] role hasCartItems
  End

- SessionManager contains some Genres
  association SessionManagerGenre between
    SessionManager[1] role inSessionManager
    Genre[1..*] role hasGenre
  End

- SessionManager contains some Users
association SessionManagerUser between
    SessionManager[1] role inSessionManager
    User[1..*] role hasUser
End

- each Order contains at least one Product
association OrderHasProducts between
    Order[0..*] role inOrder
    Product[1..*] role contains
End

- each Order belongs to exactly one Customer
association CustomerHasOrder between
    Customer[1] role belongsTo
    Order[0..*] role hasOrder
End

--
--CONSTRAINTS
--
Constraints
- (1) The userId for each User must be unique.
    context User
    inv UniqueuserId:
    User.allInstances -> forAll(u1, u2 | u1 <> u2 implies u1.userId <> u2.userId)
- (2) Products belong to exactly one genre
    context Genre
    inv ProductOneGenre:
    Genre.allInstances -> forAll(g1, g2 | g1 <> g2 && g1.hasProducts->
includes(p) implies g2.hasProducts-> excludes(p))

- (3) Price of the product should be positive
  
  context Product
  
  inv productPrice:
  
  self.Price > 0

- (4) Each Shopping cart belongs to only one customer
  
  context ShoppingCart
  
  inv CarthasOneCustomer
  
  ShoppingCart.allInstances -> forAll (s1, s2 | s1.cartId <> s2.cartId
  
  implies s1.customer <> s2.customer)

- (5) Each shoppingcart has atleast one cartItem
  
  context ShoppingCart
  
  inv CartContainsItems
  
  self.hasCartItems -> notEmpty()

- (6) The OrderId for each Order must be unique.
  
  context Order
  
  inv UniqueOrderId:
  
  Order.allInstances -> forAll(o1, o2 | o1 <> o2 implies o1.orderId <>
  
  o2.orderId)

- (7) Each Order has atleast some products
  
  context Order
  
  inv OrderContainsProducts
self.contains -> notEmpty()

- (8) Each order belongs to exactly one customer
  context Order
  inv OrdertoOneCustomer
  Order.allInstances -> forAll (o1, o2 | o1.orderId <> o2.orderId
  implies o1.belongsto <> o2.belongsto)

-- ADMINISTRATOR
- (9) An administrator's user name is unique
  context Administrator
  inv userIdUnique:
  Administrator.allInstances->forAll(a1,a2 | a1<>a2 implies a1.userId
  <> a2.userId)
Chapter 4 – Inspection List

1. **Purpose**

   The purpose of this document is to provide a checklist for the technical inspectors of the Online Music CD Store Website. The goal of the technical inspection process is to aid the developer in checking for correctness and consistency with the architectural design, which ensures the quality of the software design.

2. **Items to be Inspected**

   - Vision Document 2.0
   - Architecture Design Document 1.0
   - UML Diagrams
     - Class diagrams
     - Sequence diagrams
   - Formal Specification
   - USE Model

3. **Formal Technical Inspectors**

   - Rahul Deshmukh
   - Srungokshi Neelakantan

4. **Formal Technical Inspection Checklist**

<table>
<thead>
<tr>
<th>Item to be Inspected</th>
<th>Pass/Fail/Partial</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. The documentation follows the MSE portfolio standard which is described at <a href="http://mse.cis.ksu.edu/oncampus/mse-portfolio.htm">http://mse.cis.ksu.edu/oncampus/mse-portfolio.htm</a></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. All the classes in the USE model are represented in the class diagram.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. The multiplicities in the USE model have been depicted in the class diagram.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

79
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4.</td>
<td>All the symbols used in the class diagrams are according to the UML standards.</td>
</tr>
<tr>
<td>5.</td>
<td>All the classes in the class diagram are clear as to what they represent in the architecture design document.</td>
</tr>
<tr>
<td>6.</td>
<td>Class Diagram matches the code parameters.</td>
</tr>
<tr>
<td>7.</td>
<td>The symbols used in the sequence diagram correspond to UML standards.</td>
</tr>
<tr>
<td>8.</td>
<td>Sequence diagram matches class diagram.</td>
</tr>
<tr>
<td>9.</td>
<td>All the requirements in the Software Requirements Specification have been covered in the Architecture Design Document.</td>
</tr>
</tbody>
</table>

Table 4 Formal Technical Inspection List
Chapter 5 – Component Design

1. Introduction

The purpose of this document is to provide a component design for the Online MusicCD Store. The design will outline the internal design of each component.

2. Class Diagram

Fig.29. Middle Tier Class Diagram
2.1. Class Descriptions

2.1.1. User

This class will handle all user actions. The User class is the super class of Customer and Administrator.

| User |  
|------|---|
| -userId : string |  
| -password : string |  
| -loginStatus : string |  
| +verifyLogin(in userid : string, in password : string) : bool |  

Fig. 11. User Class Component Diagram

Attributes and Methods:

- userId: The user’s userid to login
- password: The user’s password to login
- loginStatus: The user’s login status
- verifyLogin(): Responsible for authenticating and authorizing the user to user secure sites

2.1.2. Customer

This class will handle the customer actions. It inherits all the User class responsibilities and its functions.

| Customer |  
|----------|---|
| -customerName : string |  
| -address : string |  
| -email : string |  
| -phoneNumber : int |  
| -creditCardInfo : string |  
| -shippingInfo : string |  
| +register(in userid : string, in password : string, in emailId : string) : void |  
| +login(in userid : void, in password : void) : void |  
| +updateProfile(in customerName : string, in emailId : string, in address : string) : void |  
| +search(in keyWord : string) : void |  

Fig. 31. Customer Class Component Diagram

Attributes and Methods:

- customerId: The user’s customerid
• password: The user’s password
• customerName: The user’s name
• address: The user’s address
• email: The user’s email id
• phoneno: The user’s contact number
• creditcardInfo: The user’s credit card information for payment
• shippingInfo: The user’s shipping information
• register(): Responsible for user registration
• login(): Responsible for user login
• updateProfile(): Responsible for updating user’s profile information
• search(): Responsible for searching an available MusicCD

2.1.3. Administrator

This class will handle the administrator actions. It inherits all the User class responsibilities and its functions.

![Administrator Class Component Diagram]

Attributes and Methods:

• adminId: The administrator’s Id
• password: The administrator’s password
• adminName: The administrator’s name
• email: The administrator’s email id
• createGenre(): Responsible for creating a new genre
• createProduct(): Responsible for creating a new product
• deleteGenre(): Responsible for deleting an existing genre
• deleteProduct(): Responsible for deleting an existing product
• editCatalogDetails(): Responsible for editing the Catalog details
• viewOrderHistory(): Responsible for viewing user’s order history

2.1.4. Session Manager

The SessionManager class supports the User’s required operations.

<table>
<thead>
<tr>
<th>SessionManager</th>
</tr>
</thead>
<tbody>
<tr>
<td>-userId : string</td>
</tr>
<tr>
<td>-departmentName: string</td>
</tr>
<tr>
<td>+getUser(in userId : string) : User</td>
</tr>
<tr>
<td>+getdepartment(in departmentName : string) : Department</td>
</tr>
</tbody>
</table>

Fig.33. SessionManager Class Component Diagram

Attributes and Methods:

• userId: The user’s Id
• genreId: The genre id
• getUser(): Responsible for getting user information
• getAllGenres(): Responsible for getting information of all available genres

2.1.5. Genre

This class represents a Genre

<table>
<thead>
<tr>
<th>Genre</th>
</tr>
</thead>
<tbody>
<tr>
<td>-departmentId</td>
</tr>
<tr>
<td>-name</td>
</tr>
<tr>
<td>-description</td>
</tr>
<tr>
<td>+getCategoryInDepartment(in categoryId : int) : Category</td>
</tr>
</tbody>
</table>

Fig.34. Genre Class Component Diagram

Attributes and Methods:
• genreId: The genre’s id
• genreName: The genre’s name
• description(): The genre’s description
• getAllProducts(): Responsible for getting information of all available products

2.1.6. Product

This class represents collection of products of a particular genre.

<table>
<thead>
<tr>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>-productId : int</td>
</tr>
<tr>
<td>-name : string</td>
</tr>
<tr>
<td>-description : string</td>
</tr>
<tr>
<td>-price : int</td>
</tr>
<tr>
<td>-imageFileName : string</td>
</tr>
<tr>
<td>+displayProduct(productId : int) : void</td>
</tr>
<tr>
<td>+getProductDetails(productId : int) : void</td>
</tr>
</tbody>
</table>

Fig.35. Product Class Component Diagram

Attributes and Methods:

• productId: The product’s id
• productName: The product’s name
• genreName(): The products’ genre name
• artist: The artist of the product
• description: The description of product
• price: The price of product
• imageFileName: The file name of the image for the product
• getProductDetails(): Responsible for getting details of all available products

2.1.7. ShoppingCart

This class represents a shopping cart that has all the products that are added by a customer to buy.
Fig. 36. ShoppingCart Class Component Diagram

Attributes and Methods:

- **cartId**: The shopping carts’ id
- **productId**: The id of a product contained by the shopping cart
- **quantity**: The quantity of cart items in the shopping cart
- **dateAdded**: The date when a new item is added to the cart
- **item**: The shopping cart item
- **addItem**: Responsible for adding an item to the shopping cart
- **deleteCartItem**: Responsible for deleting an item from the shopping cart
- **updateQuantity**: Responsible for updating the quantity of items in the shopping cart
- **viewCartDetails**: Responsible for viewing the shopping cart details
- **checkout**: Responsible for checking out
- **calcUnitPrice**: Responsible for calculating the price of each cart item based on its quantity
- **calcTotalPrice**: Responsible for calculating the total price of the cart items

### 2.1.8. CartItem

This class represents a cart item object.
Fig.36. CartItem Class Component Diagram

Attributes and Methods:

- cartId: The items’ cart id
- customerId: The id of the user
- name: The name of the user
- productId: The id of the product in the shopping cart
- quantity: The quantity of cart items in the shopping cart
- unitCost: The cost of the cart item
- subtotal: The cost of the cart item based on its quantity

2.1.9. Order

This class will store all information regarding the orders made by each customer.

Fig.37. Order Class Component Diagram

Attributes and Methods:

- orderId: The order id
- userId: The id of the user to whom the order belongs
• dateCreated: The date on which the order is created
• dateShipped: The date on which the order is shipped
• status: The status of the order
• shippingId: The shipping id of the order
• placeOrder(): Responsible for placing the order

2.1.10. OrderDetail

This class represents the detail of each order

![OrderDetail class component diagram]

Attributes and Methods:

• orderId: The order id
• productId: The id of the product contained in the order
• productName: The name of product contained in the order
• quantity: The quantity of products in the order
• unitCost: The cost of each product in the order
• subtotal: The total cost of the order

2.1.11. ShippingInfo

This class represents the shipping information of each order.

![ShippingInfo class component diagram]
Fig. 39. ShippingInfo Class Component Diagram

Attributes and Methods:

- shippingId: The shipping id of the order
- orderId: The order id
- shippingType: The type of shipping for the order
- shippingCost: The cost of shipping
- shippingRegionId: The id of the shipping region for the order
- updateShippingInfo(): Responsible for updating the shipping information of the order

2.1.12. Keyword

This class represents a collection of all the keywords used for the searching.

<table>
<thead>
<tr>
<th>keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>-keyword : string</td>
</tr>
<tr>
<td>+findKeyword(in keyword : string) : keyword</td>
</tr>
</tbody>
</table>

Fig. 40. keyword Class Component Diagram

Attributes and Methods:

- keyword: The keyword used for searching
- findKeyword(): Responsible for matching the keyword from the database

2.1.13. searchFacade

This class has methods to get the search results according to the options chosen by the user.

<table>
<thead>
<tr>
<th>searchFacade</th>
</tr>
</thead>
<tbody>
<tr>
<td>+getByArtist(in artist : string) : productSet</td>
</tr>
<tr>
<td>+getByProductName() : productSet</td>
</tr>
<tr>
<td>+getByKeyWord(in keyword : string) : productSet</td>
</tr>
</tbody>
</table>

Fig. 41. searchFacade Class Component Diagram
Attributes and Methods:

- getbyartist(): Responsible for getting the product information based on artist
- getbyproductname(): Responsible for getting the product information based on product name
- getbykeyword(): Responsible for getting the product information based on keywords

2.1.14. RecommendationSet

This class represents a collection of products recommended for the product selected by the user.

```
RecommendationSet
-productId : int
+getrecs(in productId : int) : productSet
```

Fig.42. RecommendationSet Class Component Diagram

Attributes and Methods:

- productId: The id of the recommended product
- getrecs(): Responsible for getting product recommendations

2.1.15. ProductSet

This class represents a collection of all the products that were searched by the user.

```
productSet
-setId : int
-productId : int
+MatchtoProduct(in keyword : string) : productSet
```

Fig.43. productSet Class Component Diagram

Attributes and Methods:

- setId: The id of the set of products
- productId: The id of the product in the set
• MatchtoProduct(): Responsible for matching the user’s text with the product information
Chapter 6 – Software Quality Assurance Plan

1. Purpose

The purpose of this Software Quality Assurance Plan (SQAP) is to define the techniques, procedures, and methodologies that will be used in the project to assure timely delivery of the software that meets specified requirements within project resources. The use of this plan will help assure the following:

1) That software development, evaluation and acceptance standards are developed, documented and followed.
2) That the results of software quality review and audits will be given to appropriate management within project. This provides feedback as to how well the development effort is conforming to various development standards.
3) That test results adhere to acceptance standards.

2. Reference Documents

- Vision Document 1.0
- Project Plan 1.0

3. Management

3.1. Organization

The organization consists of supervisory committee, major professor, developer and formal technical inspectors.

**Supervisory Committee:**

Dr. Daniel Andresen  
Dr. William Hankley  
Dr. Torben Amtoft

**Major Professor:**

Dr. Daniel Andresen
**Developer:**
Reshma Sawant

**Formal Technical Inspectors:**
Rahul Deshmukh  
Srunokshi Neelkantan

### 3.2. Responsibilities

**Supervisory Committee**
The committee will be responsible for attending the presentations and reviews at the end of each phase. After each presentation, the committee will provide feedback and suggestions regarding the project.

**Major Professor**
The major professor will be responsible for supervisory committee duties as well as supervise and evaluate the work & progress done by the developer on a weekly basis.

**Developer**
The developer will be responsible for all the documentation and software development tasks of the project. The project plan will describe all the tasks to be completed by the developer under the major professor supervising. Also, the developer will meet the major professor on a weekly basis to report progress.

**Formal Technical Inspectors**
The formal technical inspectors will be responsible for a technical review of the architecture design artifacts and the formal requirements specifications and will also be required to submit a formal report based on their findings.

The technical inspectors will be provided with a “Formal Inspection Checklist” which will specify the items to be inspected. After inspection, each of the two technical inspectors will provide a report on the result of their inspection, which
will contain a cover letter and the annotated checklist. These reports will become part of the project documentation.

3.3. Tasks

All tasks that will be performed are documented in the Project Plan 1.0. This will be reviewed after the first phase to incorporate any changes. A Gantt chart will be included, which will provide a schedule for each task.

4. Software Documentation

SQA will review all project deliverable software documentation including software plans. Review checklists will be used to review these documents. These reviews will help ensure that documentation is in compliance with generated plans and procedures.

The essential documentation of a system’s architecture and implementation will include the following:

- Vision document, project plan, software quality assurance plan, formal requirements specification, architecture design, test plan, formal technical inspection, prototype, user manual, component design, source code, assessment evaluation, project evaluation, references, and formal technical inspection letters. The supervisory committee will review all documentation for final approval.

All documentation will be posted on the developer’s web site at:

http://people.cis.ksu.edu/~reshma/project.htm

5. Standards, Practices, Conventions, and Metrics

Documentation Standard

The IEEE standards will be used as a guideline for all documentation of the project.

Coding Standard
The project will follow the guidelines in the C# coding standards and style guide.

**Commentary Standards**

- Comments must be used in the project to give a brief description of the code, focusing on the functionality and purpose of commented areas.
- Each block of statement must be well commented.
- Each routine must have a comment consisting of one or two lines, which should be placed above that particular routine, describing the purpose and limitations of the said routine.
- Each file, module, and program must contain the author’s name, date it was written or last modified, and a brief description of how the said code fits into the end product. This may or may not include external functions that are referenced within the custom code.

**Metrics**

Basic COCOMO will be used to estimate the project time and effort.

6. **Reviews and Audits**

   The purpose of reviews and audits is to specify that the evidence of work generated is adequate to ensure compliance with project and contract requirements. Audits will occur at the end of each phase. The reviews will be conducted by the supervisory committee prior to any baseline release of executable code. The reviews will ensure that:

   1) The code has been tested and meets module specifications.
   2) That any changes to applicable software module design documents have been identified.
   3) That appropriate validation tests have been run.
   4) That the functionality of the baseline is documented.
   5) That all software design documentation complies with the plans and procedures.
6) That tools and techniques used to produce and validate the Software System are identified and controlled.

The two formal technical inspectors will assess the architecture design artifacts and submit a formal report based on their findings.

7. Test and Problem Reporting

A Software Test Plan (STP) will be written to satisfy the requirements. The plan will provide management and the testing function with an overview of the test activities, schedules and resources required to perform testing. The plan will describe how the testing specifications will be implemented. Unresolved problems will be reported directly to the supervisory committee.

8. Tools, Techniques, and Methodologies

The following tools will be used for coding, testing, and documentation:

- Microsoft Visual Studio .NET 2005 IDE – for coding
- C# - for coding
- JavaScript – for coding
- HTML – for coding
- XML – for coding
- CSS – for coding
- IIS – for web server
- ASP.NET – for web forms
- MS SQL – for database server
- MS Word 2007 – for documentation
- MS Project 2007 – Project Planning
- USE 2.0.1 – for formal specification and verification
- NUnit – for unit testing
- JMeter – for performance testing
9. Records collection, Maintenance, and Retention

Three copies of the design documentation will be produced. One will be kept in the University library, one with the Major Professor, and one with the developer. The entire source code, documentation, and web pages for the project website will also be submitted to the Major Professor in the form of a CD. A CD copy should also be kept with the developer.

10. Deliverables

The following set of deliverables will be submitted at the end of each phase:

Phase I
- Vision Document
- Project Plan
- Software Quality Assurance Plan
- Demonstration

Phase II
- Action Items –identified during phase I
- Vision Document
- Project Plan
- Formal Requirements Specification
- Architecture Design
- Test Plan
- Formal Technical Inspection
- Executable Architecture Prototype

Phase III
- Action Items - identified during phase II
- User Manual
- Component Design
- Source Code
- Assessment Evaluation
• Project Evaluation
• References
• Formal Technical Inspection Letters
Chapter 7 – Test Plan

1. Introduction

This document will include the plan for testing the critical use cases and functionalities described in the Vision Document.

1.1 Objectives

The test plan for the system should support following objectives:

- Identify which features of the system will be tested.
- Define the pass/fail criteria for each feature to be tested.
- Specify the testing approaches that will be used during testing.
- Identify the deliverables of the testing process.

2. Features to be tested

This section outlines all the features that will be tested:

<table>
<thead>
<tr>
<th>Feature Identifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER</td>
<td></td>
</tr>
<tr>
<td>T-01</td>
<td>System Register</td>
</tr>
<tr>
<td>T-02</td>
<td>System Login</td>
</tr>
<tr>
<td>T-03</td>
<td>Product Search</td>
</tr>
<tr>
<td>T-04</td>
<td>Add to Cart</td>
</tr>
<tr>
<td>T-05</td>
<td>Edit Shopping Cart</td>
</tr>
<tr>
<td>T-06</td>
<td>Place Order</td>
</tr>
<tr>
<td>ADMINISTRATOR</td>
<td></td>
</tr>
<tr>
<td>T-07</td>
<td>System Login</td>
</tr>
<tr>
<td>T-08</td>
<td>Create and Delete product from a Category</td>
</tr>
<tr>
<td>T-09</td>
<td>Create and Delete Category from a Genre</td>
</tr>
<tr>
<td>T-10</td>
<td>Create and Delete Genre from the Catalog</td>
</tr>
<tr>
<td>T-11</td>
<td>Manage Orders</td>
</tr>
<tr>
<td>T-12</td>
<td>Manage Shopping Carts</td>
</tr>
</tbody>
</table>

Table 6 Features to be Tested
3. Test Cases for User Requirements

3.1. Register

3.1.1. Test 1:

<table>
<thead>
<tr>
<th>Purpose:</th>
<th>Test that users can register with the proper username and password.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incorrect Input:</td>
<td>Username already in use with other existing users.</td>
</tr>
<tr>
<td>Pass criteria:</td>
<td>An appropriate message should be displayed and the user should not be allowed to register for an account with that username.</td>
</tr>
<tr>
<td>Correct Input:</td>
<td>Username not in use with other existing users.</td>
</tr>
<tr>
<td>Pass criteria:</td>
<td>User should be able to register with the website and directed to the secure Web page requested.</td>
</tr>
</tbody>
</table>

Steps:
1. visit Register.aspx Web page
2. enter username
3. enter password, confirm password, Email, Security question and Security answer
4. click Signup button
5. If the username already in use, check if the message is to re-enter another username.

3.1.2. Test 2:

<table>
<thead>
<tr>
<th>Purpose:</th>
<th>Test that users can register with the proper username and password.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incorrect Input:</td>
<td>“Password” and “Confirm password” fields do not match.</td>
</tr>
<tr>
<td>Pass criteria:</td>
<td>An appropriate message should be displayed and the user should not be allowed to register for a new account until both the fields match.</td>
</tr>
<tr>
<td>Correct Input:</td>
<td>“Password” and “Confirm password” fields match.</td>
</tr>
<tr>
<td>Pass criteria:</td>
<td>User should be able to register with the website and directed to the secure Web page requested.</td>
</tr>
</tbody>
</table>
### 3.1.3. Test 3:

<table>
<thead>
<tr>
<th>Purpose:</th>
<th>Test that users can register with the proper username and password.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incorrect Input:</td>
<td>An empty required field. (Username, password, confirm password, Email, Security question and Security answer)</td>
</tr>
<tr>
<td>Pass criteria:</td>
<td>An appropriate message should be displayed and the user should not be allowed to register for a new account.</td>
</tr>
<tr>
<td>Correct Input:</td>
<td>All fields entered correctly.</td>
</tr>
<tr>
<td>Pass criteria:</td>
<td>User should be able to register with the website and directed to the secure Web page requested.</td>
</tr>
</tbody>
</table>
| Steps: | 1. visit Register.aspx Web page  
2. Click Signup button with one of the fields left blank  
3. Check if the message is to enter all required fields. |

### 3.2. System Login

#### 3.2.1. Test 1:

<table>
<thead>
<tr>
<th>Purpose:</th>
<th>Test that users can login with the proper username and password.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incorrect Input:</td>
<td>Wrong username and/or password (the combination of username and password that does not match with the one in the database).</td>
</tr>
<tr>
<td>Pass criteria:</td>
<td>An appropriate message should be displayed and the user should not be allowed to login.</td>
</tr>
<tr>
<td>Correct Input:</td>
<td>Right username and password.</td>
</tr>
</tbody>
</table>
3.2.2. Test 2:

<table>
<thead>
<tr>
<th>Purpose:</th>
<th>Test that users can login with the proper username and password.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incorrect Input:</td>
<td>An empty required field. (username or password)</td>
</tr>
<tr>
<td>Pass criteria:</td>
<td>An appropriate message should be displayed and the user should not be allowed to login.</td>
</tr>
<tr>
<td>Correct Input:</td>
<td>Right username and password.</td>
</tr>
<tr>
<td>Pass criteria:</td>
<td>User should be able to login on the website and directed to the secure Web page requested.</td>
</tr>
</tbody>
</table>
| Steps: | 1. visit Login.aspx Web page  
2. click login button with one of the fields is empty  
3. Check if the message is to enter all required fields. |

3.3. Product Search

<table>
<thead>
<tr>
<th>Purpose:</th>
<th>Test that after entering the keyword, requested and relevant products are returned.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incorrect Input:</td>
<td>Text that does not match any of product names and/or description.</td>
</tr>
<tr>
<td>Pass criteria:</td>
<td>An appropriate message should be displayed that no search results found.</td>
</tr>
<tr>
<td>Correct Input:</td>
<td>Text that matches one or more of product names and/or description.</td>
</tr>
<tr>
<td>Pass criteria:</td>
<td>The search results that match the input text will be</td>
</tr>
</tbody>
</table>
1. Enter the inputs in the text box against Song Name, Artist, Style and Year
2. Click on search button
3. Check whether the products that have attributes matching with these inputs are displayed, if not appropriate message is displayed.

### 3.4. Edit Shopping Cart

| Purpose: | Test that clicking Update Quantities will update the cart summary accordingly. |
| Incorrect Input: | Negative input number or input other than integer number in “Quantity” field |
| Pass criteria: | An appropriate message should be displayed. |
| Correct Input: | Positive integer number. |
| Pass criteria: | The product quantity should be updated or deleted according to the specified input number. |

| Steps: |
| 1. visit ShoppingCart.aspx Web page |
| 2. Check after entering incorrect input, an appropriate message should be displayed. |
| 3. If entered a valid number, check if the total quantity and relative price is updated after clicking update or delete button. |

### 3.5 Place Order

| Purpose: | Test that user can place an order with valid profile information. |
| Incorrect Input: | An empty required field. (First name and last name, Street address, city, state, zip code, email, credit card information, shipping address) |
| Pass criteria: | An appropriate message should be displayed and the user should not be allowed to place the order. |
| Correct Input: | The user should be able to place an order and redirected |
to the order confirmation page.

<table>
<thead>
<tr>
<th>Pass criteria:</th>
<th>The product quantity should be updated or deleted according to the specified input number.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Steps:</td>
<td>1. visit Checkout.aspx Web page</td>
</tr>
<tr>
<td></td>
<td>2. Click Place Order button with one of the fields empty.</td>
</tr>
<tr>
<td></td>
<td>3. Check if the message is to enter all required fields.</td>
</tr>
</tbody>
</table>

3.6. **Add to Cart**

<table>
<thead>
<tr>
<th>Purpose:</th>
<th>Test that clicking add to cart button, product is getting added in the cart.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input:</td>
<td>User selects product and clicks Add to cart button.</td>
</tr>
<tr>
<td>Pass criteria:</td>
<td>If Cart is empty or the product selected is not already in the cart before the action, the product gets added to it. If product already exists then its quantity is updated by 1</td>
</tr>
<tr>
<td>Steps:</td>
<td>1. Click Add to Cart button</td>
</tr>
<tr>
<td></td>
<td>2. Check whether the cart shows the product</td>
</tr>
<tr>
<td></td>
<td>3. Check if the quantity count is one, if product is not already in the cart.</td>
</tr>
<tr>
<td></td>
<td>4. Check if the quantity count is increased by one, if product is already in the cart.</td>
</tr>
</tbody>
</table>

4. **Test Cases for Administrator Requirements**

4.1. **System Login**

<table>
<thead>
<tr>
<th>Purpose:</th>
<th>Test that only administrator is able to login and have privileges to change the catalog information.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incorrect Input:</td>
<td>Invalid username and/or password, Empty required field</td>
</tr>
<tr>
<td>Pass criteria:</td>
<td>An appropriate message should be displayed and the user should not be allowed to login.</td>
</tr>
<tr>
<td>Correct Input:</td>
<td>Valid username and password, All fields entered correctly</td>
</tr>
<tr>
<td>Pass criteria:</td>
<td>User should be able to login on the website and directed to the secure Web page requested.</td>
</tr>
</tbody>
</table>
### Steps:

1. visit Login.aspx Web page  
2. Enter username  
3. Enter password  
4. Click login button  
5. Check if the message is to re-enter username or password  
6. Or click login button with one of the fields empty  
7. Check if the message is to enter all the required fields.

---

### 4.2. Delete a Product from Category

<table>
<thead>
<tr>
<th>Purpose:</th>
<th>Test that after clicking Delete Product, selected product is being removed from the catalog.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input:</td>
<td>Administrator selects a product to be deleted, the category from which it is to be deleted and clicks delete button.</td>
</tr>
<tr>
<td>Pass criteria:</td>
<td>Product is successfully removed from that category but not from others, if it belongs to more than one category.</td>
</tr>
</tbody>
</table>
| Steps:   | 1. Click Delete Product button  
2. Check for the message- product has been removed from the category.  
3. Check that after clicking Show products in category, the deleted product is not shown. |

---

### 4.3. Delete Category from a Genre

<table>
<thead>
<tr>
<th>Purpose:</th>
<th>Test that after clicking Delete category, selected category and its associated products are being removed from the genre.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input:</td>
<td>Administrator selects a category to be deleted and clicks delete button.</td>
</tr>
<tr>
<td>Pass criteria:</td>
<td>Category is successfully removed from that genre.</td>
</tr>
</tbody>
</table>
| Steps:   | 1. Click Delete Category button  
2. Check for the message- category and its associated product is removed from the genre.  
3. Check that after clicking Show categories in genre, the deleted category is not shown. |
### 4.4. Delete Genre from the Catalog

<table>
<thead>
<tr>
<th>Purpose:</th>
<th>Test that after clicking Delete genre, selected genre, its associated categories and their associated products are being removed from the database.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input:</td>
<td>Administrator selects a genre to be deleted and clicks delete button.</td>
</tr>
<tr>
<td>Pass criteria:</td>
<td>The selected genre is successfully removed from the catalog.</td>
</tr>
</tbody>
</table>
| Steps: | 1. Click Delete Genre button  
2. Check for the message- genre, its associated category and their associated products are removed from the catalog.  
3. Check that on the main page the deleted department does not appear. |

### 4.5. Manage Shopping Carts

<table>
<thead>
<tr>
<th>Purpose:</th>
<th>Test that after clicking Delete old shopping carts, carts before and up to that date are removed.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input:</td>
<td>Administrator enters the date range for which the shopping carts are to be removed and clicks delete Old Shopping Carts button.</td>
</tr>
<tr>
<td>Pass criteria:</td>
<td>The shopping carts that were not modified within that range are removed from the database but carts after that date should exist.</td>
</tr>
</tbody>
</table>
| Steps: | 1. Select the days from drop down list  
2. Click Delete Old Shopping Carts  
3. Check for the message- carts are successfully removed  
4. Check that carts before or up to that date does not exist in database. |

### 4.6. Manage Orders

| Purpose: | Test that order status is correctly updated. |
5. **Approach**

Only functional black box testing will be performed to test the functionality of the system. The features mentioned above describe how the user will interact with the system, so the testing will require the tester to interact with the system in the same way a typical user would. The user actions will be simulated through a set of test scenarios. Each scenario will trace back to a requirement listed in the Vision Document.

5.1. **NUnit Testing**

In NUnit testing each module is tested alone in an attempt to discover any errors in the code, i.e. all executable source code will be subject to test. It verifies that certain parts of the code are working properly. NUnit testing tool will be used to perform the unit testing.

5.2. **Performance Testing**

This will be performed to test the entire system to see whether all driving requirements are satisfied. Allowing multiple users log into the system and perform the operations at the same time using the JMeter testing tool will do this. This test verifies that the components of the systems meet the stated requirements for speed.
Using JMeter tool, approximately 100 concurrent virtual users with a minimum of 50 requests per user will be inputted to calculate the response times for each of the above components. The above tests would be done in different environments like:

- Local connection of 36.0 Mbps
- LAN connection of 100.0 Mbps
- Machine to machine testing

5.3. Load Testing

A system that works well with one user may completely break down with several hundred or more users. Load testing helps to evaluate how well a system holds up under light, medium and heavy use.

6. Item Pass/Fail Criteria

Test cases executed on the Online Music Store will pass if they meet the specific requirements as mentioned in the Vision Document. A test case will fail if any behavioral expectation is not met as described.

7. Suspension Criteria and Resumption Requirements

7.1. Suspension Criteria

If a test case fails, testing will be suspended for all dependent features. The failed test case will be logged into a test log along with a description of the failure.

7.2. Resumption Requirement

Test cases, not dependent on the case in which a bug is reported, will continue to be executed in parallel to bug fixing. Testing for the failed test case will resume after the bug has been identified and resolved.
8. Test Deliverables

The following artifacts will be produced after the tests are conducted on the system:

- Test Plan
- Test Cases
- Test Log

9. Environmental Needs

9.1. Hardware

All tests will be conducted on an Intel-based machine.

9.2. Software

Microsoft Visual Studio 2005, NUnit, and JMeter testing software will be used as the testing environment.

9.3. Operating System

All tests to be performed will run under the Windows XP Professional platform.
Chapter 8 – Assessment Evaluation

1. Introduction

This document presents the results of the functional testing performed on Online Music Store. The test cases are in reference to the test cases defined in the Test Plan 1.0 from Phase II.

2. Testing Results Summary

<table>
<thead>
<tr>
<th>Test Case #</th>
<th>Description</th>
<th>Results/Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER</td>
<td>T-01 System Register</td>
<td>Passed</td>
</tr>
<tr>
<td></td>
<td>T-02 System Login</td>
<td>Passed</td>
</tr>
<tr>
<td></td>
<td>T-03 Add to Cart</td>
<td>Passed</td>
</tr>
<tr>
<td></td>
<td>T-04 Edit Shopping Cart</td>
<td>Passed</td>
</tr>
<tr>
<td></td>
<td>T-05 Place Order</td>
<td>Passed</td>
</tr>
<tr>
<td>ADMINISTRATOR</td>
<td>T-06 Create and Delete product from Category</td>
<td>Passed</td>
</tr>
<tr>
<td></td>
<td>T-07 Create and Delete Category from a Genre</td>
<td>Passed</td>
</tr>
<tr>
<td></td>
<td>T-08 Create and Delete Genre from Catalog</td>
<td>Passed</td>
</tr>
<tr>
<td></td>
<td>T-09 Manage Orders</td>
<td>Passed</td>
</tr>
<tr>
<td></td>
<td>T-10 Manage Shopping Carts</td>
<td>Passed</td>
</tr>
</tbody>
</table>

Table 5 Test Case Result Summary

3. Detailed Test Results

3.1. Manual Testing

Manual Testing of the system was done to ensure the correctness of various parts of the code.

3.1.1. Test Case 01 – Register Web Form
<table>
<thead>
<tr>
<th>Test Unit</th>
<th>Test Case</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>btnSignup</td>
<td>An empty requirement field (Username, password, confirm password, Email, Security question and Security answer)</td>
<td>System prompts user with a message “All fields are required. Please try again”.</td>
</tr>
<tr>
<td></td>
<td>Username already in use with other existing users.</td>
<td>System prompts user to enter the username with a message “Please enter a different username”</td>
</tr>
<tr>
<td></td>
<td>Password and Confirm password fields do not match.</td>
<td>System prompts user to enter the password with a message “The Password and Confirmation Password must match”.</td>
</tr>
<tr>
<td></td>
<td>All valid requirement fields entered</td>
<td>System redirected the user to the secure Login Web page.</td>
</tr>
</tbody>
</table>

Table 6 System Register Test Case Result Summary

3.1.2. Test Case 02 – Login Web Form for User

<table>
<thead>
<tr>
<th>Test Unit</th>
<th>Test Case</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>btnLogin</td>
<td>An empty requirement field (Username and/or password)</td>
<td>System prompts user to enter username and password with a message “Please enter username” or “Please enter password”</td>
</tr>
<tr>
<td></td>
<td>An invalid username and/or password entered</td>
<td>System prompts user to enter the username and password with a message “Invalid username and/or password. Please try again”.</td>
</tr>
<tr>
<td></td>
<td>A valid username and/or password entered</td>
<td>System redirected the user to the secure Web page.</td>
</tr>
</tbody>
</table>

Table 9 System Login Test Case Result Summary
### 3.1.3. Test Case 03 – Product search

<table>
<thead>
<tr>
<th>Test Unit</th>
<th>Test Case</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>btnSearch</td>
<td>Valid inputs and valid selection from each category (Style, Format and Price range)</td>
<td>System displayed the matched results of the search made otherwise displayed the message “The text did not match any results”.</td>
</tr>
</tbody>
</table>

Table 10 System Search Test Case Result Summary

### 3.1.4. Test case 04 – Add to Cart

<table>
<thead>
<tr>
<th>Test Unit</th>
<th>Test Case</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>btnAddtocart</td>
<td>Product is selected and Addtocart button is clicked</td>
<td>If Cart is empty or the product selected is not already in the cart before the action, the product gets added to it. If product already exists then its quantity is updated by 1</td>
</tr>
</tbody>
</table>

Table 11 System Add to Cart Test Case Result Summary

### 3.1.5. Test Case 05 - Edit Shopping Cart

<table>
<thead>
<tr>
<th>Test Unit</th>
<th>Test Case</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>btnUpdate,</td>
<td>Negative input number or input other than integer number entered in “Quantity” field</td>
<td>System prompts the user with the message “Please enter a valid number”.</td>
</tr>
<tr>
<td>btnDelete</td>
<td>Valid Positive number entered in “Quantity” field</td>
<td>System updates the product quantity and displays the message “Your shopping cart was successfully updated” or “Item successfully deleted”</td>
</tr>
</tbody>
</table>

Table 12 System Edit Shopping Cart Test Case Result Summary

### 3.1.6. Test Case 06 – Place Order
### 3.1.7. Test Case 07 – Login Web form for Administrator

<table>
<thead>
<tr>
<th>Test Unit</th>
<th>Test Case</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>btnLogin</td>
<td>An empty requirement field (Username and/or password)</td>
<td>System prompts administrator to enter username and password with a message “Please enter username” or “Please enter password”</td>
</tr>
<tr>
<td></td>
<td>An invalid username and/or password entered</td>
<td>System prompts administrator to enter the username and password with a message “Invalid username and/or password. Please try again”.</td>
</tr>
<tr>
<td></td>
<td>A valid username and/or password entered</td>
<td>System redirected the administrator to the secure Web page.</td>
</tr>
</tbody>
</table>

Table 14 System Login Test Case Result Summary

### 3.1.8. Test Case 08 – Create and Delete product from a Category

<table>
<thead>
<tr>
<th>Test Unit</th>
<th>Test Case</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>btnCreateProduct</td>
<td>An empty requirement field (Name, description, Price and Image File)</td>
<td>System prompts user to enter all the requirement fields with a message “All fields are required”</td>
</tr>
</tbody>
</table>
114

<table>
<thead>
<tr>
<th>Test Unit</th>
<th>Test Case</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>btnCreateCategory</td>
<td>An empty requirement field (Name and description)</td>
<td>System prompts user to enter Name and description with a message “All fields are required”</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A new category is created and added to the database.</td>
</tr>
<tr>
<td>btnDeleteCategory</td>
<td>Select the category and its respective genre from which to delete.</td>
<td>System deleted the category from the respective genre and displays the message “Category removed successfully”</td>
</tr>
</tbody>
</table>

Table 16 System Create & Delete Category Test Case Result Summary

3.1.10. Test Case 10 – Create and Delete Genre from Catalog

<table>
<thead>
<tr>
<th>Test Unit</th>
<th>Test Case</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>btnCreateGenre</td>
<td>An empty requirement field (Name and description)</td>
<td>System prompts user to enter Name and description with a message “All fields are required”</td>
</tr>
</tbody>
</table>

Table 15 System Create & Delete Product Test Case Result Summary

3.1.9. Test Case 09 - Create and Delete Category from a Genre

<table>
<thead>
<tr>
<th>Test Unit</th>
<th>Test Case</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>btnDeleteProduct</td>
<td>Select the product and its respective category from which to delete.</td>
<td>System deleted the product from the respective category and displays the message “Product removed successfully”</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A new product is created and added to the database.</td>
</tr>
</tbody>
</table>

Table 15 System Create & Delete Product Test Case Result Summary
<table>
<thead>
<tr>
<th>Test Unit</th>
<th>Test Case</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>goButton</td>
<td>Negative input number to view recent orders and invalid end dates (date greater than the system date)</td>
<td>System prompts the administrator to enter the valid number for viewing orders and dates with the message “Please enter a valid number” and “Please enter valid end dates”.</td>
</tr>
<tr>
<td></td>
<td>Valid Positive number and valid end dates entered</td>
<td>System displays the orders to the administrator.</td>
</tr>
</tbody>
</table>

Table 17 System Manage Orders Test Case Result Summary

3.1.12. Manage Shopping Carts

<table>
<thead>
<tr>
<th>Test Unit</th>
<th>Test Case</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>goCount</td>
<td>Select one of the options from All shopping carts, One, Ten, Twenty, Thirty, ninety as the days for which to count the number of shopping carts</td>
<td>System displays the number of shopping carts existing for the days selected otherwise displays the message “There are no old shopping carts”</td>
</tr>
<tr>
<td>goDelete</td>
<td>Select one of the options from All shopping carts, One, Ten, Twenty, Thirty, ninety as the days for which to delete the number of shopping carts</td>
<td>System deleted the number of shopping carts existing for the days selected and displays the message “Old shopping carts successfully deleted”</td>
</tr>
</tbody>
</table>
3.2. Performance Testing

The tool, Jakarta JMeter, was used to simulate the virtual users (clients) and test the performance of the system. **Apache JMeter** is a 100% pure Java desktop application designed to load test client/server software (such as a web application). It may be used to test performance both on static and dynamic resources such as static files, Java Servlets, CGI scripts, Java objects, databases, FTP servers, and more. JMeter can be used to simulate a heavy load on a server, network or object to test its strength or to analyze overall performance under different load types.

For performance testing of an application, there are number of questions to be asked first:

1) **What is our anticipated peak number of users?**

   The hardware capabilities of my laptop limit the number of threads to 1000 that can run effectively with JMeter. It will also depend on how fast the server on the laptop is (a faster server makes JMeter work harder since it returns request quicker). The more JMeter works, the less accurate its timing information will be. The more work JMeter does, the more each thread has to wait to get access to the CPU, the more inflated the timing information gets.

2) **When is a good time to load-test our application (i.e. off-hours or week-ends)?**

   The best time to load-test the application is during the weekend, but due to time constraint it has to be tested during the working hours.

3) **What is the testing intended to achieve?**

   Here, performance testing is intended to achieve an estimate of the peak and sustained load the application can handle for Wireless and LAN connection.
To test the application for peak load, a web page from each of the following category is selected and tested for wireless connection and LAN connection:

- HTML Page (Login Web Page)
- Database Intensive Page (Home Page)
- Business Logic Page (Shopping Cart Page)

3.1.1. Wireless Connection at 54.0 Mbps for Peak Load

3.1.1.1. Login Web Page [http://localhost:2416/CDShop/Login.aspx]

The response time for different combinations of users and requests is noted.

(i) In the following test, the number of loop-count is kept constant at 20,000 and the users are increased at intervals. The maximum number of users is limited to 1000 due to the hardware capabilities. A graph is plot for number of requests (X-axis) versus the response time (Y-axis).

<table>
<thead>
<tr>
<th>Users Count</th>
<th>Loop Count</th>
<th>Ramp-up period (sec)</th>
<th>Average Response Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>20000</td>
<td>5</td>
<td>1900</td>
</tr>
<tr>
<td>600</td>
<td>20000</td>
<td>5</td>
<td>7439</td>
</tr>
<tr>
<td>800</td>
<td>20000</td>
<td>5</td>
<td>8500</td>
</tr>
<tr>
<td>1000</td>
<td>20000</td>
<td>5</td>
<td>13000</td>
</tr>
</tbody>
</table>
The above graph shows a steep increase in the response time as the number of users increased.

(ii) In the following test, the number of users is kept constant and the number of requests is increased and a graph is plot for number of requests (X-axis) versus the response time (Y-axis).

<table>
<thead>
<tr>
<th>Users</th>
<th>Loop Count</th>
<th>Ramp-up period (sec)</th>
<th>Number of Requests</th>
<th>Average Response Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>100</td>
<td>5</td>
<td>5000</td>
<td>877</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>5</td>
<td>10,000</td>
<td>995</td>
</tr>
<tr>
<td>100</td>
<td>200</td>
<td>5</td>
<td>20,000</td>
<td>1001</td>
</tr>
<tr>
<td>100</td>
<td>400</td>
<td>5</td>
<td>40,000</td>
<td>1030</td>
</tr>
<tr>
<td>100</td>
<td>600</td>
<td>5</td>
<td>60000</td>
<td>1043</td>
</tr>
<tr>
<td>100</td>
<td>1000</td>
<td>5</td>
<td>100000</td>
<td>1060</td>
</tr>
</tbody>
</table>

The response time for different combinations of users and requests is noted.

(i) In the following test, the number of loop-count is kept constant at 20,000 and the users are increased at intervals. The maximum number of users is limited to 1000 due to the hardware capabilities. A graph is plot for number of requests (X-axis) versus the response time (Y-axis).

<table>
<thead>
<tr>
<th>Users</th>
<th>Loop Count</th>
<th>Ramp-up period (sec)</th>
<th>Average Response Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>20000</td>
<td>5</td>
<td>8354</td>
</tr>
<tr>
<td>600</td>
<td>20000</td>
<td>5</td>
<td>22538</td>
</tr>
<tr>
<td>800</td>
<td>20000</td>
<td>5</td>
<td>29567</td>
</tr>
<tr>
<td>1000</td>
<td>20000</td>
<td>5</td>
<td>38603</td>
</tr>
</tbody>
</table>
In the following test, the number of users is kept constant and the number of requests is increased and a graph is plot for number of requests (X-axis) versus the response time (Y-axis).

<table>
<thead>
<tr>
<th>Users</th>
<th>Loop Count</th>
<th>Ramp-up period (sec)</th>
<th>Number of Requests</th>
<th>Average Response Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>100</td>
<td>5</td>
<td>5000</td>
<td>1873</td>
</tr>
<tr>
<td>100</td>
<td>200</td>
<td>5</td>
<td>20,000</td>
<td>3701</td>
</tr>
<tr>
<td>100</td>
<td>400</td>
<td>5</td>
<td>40,000</td>
<td>3785</td>
</tr>
<tr>
<td>100</td>
<td>600</td>
<td>5</td>
<td>60000</td>
<td>3827</td>
</tr>
<tr>
<td>100</td>
<td>800</td>
<td>5</td>
<td>80000</td>
<td>3837</td>
</tr>
</tbody>
</table>
(iii) In the following test, the number of users is kept constant at 200 and the number of requests is increased up to 20,000 and a graph is plot for number of requests (X-axis) versus the response time (Y-axis).

<table>
<thead>
<tr>
<th>Users</th>
<th>Loop Count</th>
<th>Ramp-up period (sec)</th>
<th>Number of Requests</th>
<th>Average Response Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>1000</td>
<td>5</td>
<td>200000</td>
<td>7936</td>
</tr>
<tr>
<td>200</td>
<td>5000</td>
<td>5</td>
<td>1000000</td>
<td>8300</td>
</tr>
<tr>
<td>200</td>
<td>10000</td>
<td>5</td>
<td>2000000</td>
<td>8440</td>
</tr>
<tr>
<td>200</td>
<td>15000</td>
<td>5</td>
<td>3000000</td>
<td>8663</td>
</tr>
<tr>
<td>200</td>
<td>20000</td>
<td>5</td>
<td>4000000</td>
<td>9036</td>
</tr>
</tbody>
</table>


The response time for different combinations of users and requests is noted.

(i) In the following test, the number of loop-count is kept constant at 20,000 and the users are increased at intervals. The maximum number of users is limited to 1000 due to the hardware capabilities. A graph is plot for number of requests (X-axis) versus the response time (Y-axis).
(i) In the following test, the number of users is kept constant and the number of requests is increased and a graph is plot for number of requests (X-axis) versus the response time (Y-axis).

<table>
<thead>
<tr>
<th>Users</th>
<th>Loop Count</th>
<th>Ramp-up period (sec)</th>
<th>Number of Requests</th>
<th>Average Response Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>100</td>
<td>5</td>
<td>5000</td>
<td>750</td>
</tr>
<tr>
<td>100</td>
<td>200</td>
<td>5</td>
<td>20,000</td>
<td>1205</td>
</tr>
<tr>
<td>100</td>
<td>400</td>
<td>5</td>
<td>40,000</td>
<td>1205</td>
</tr>
<tr>
<td>100</td>
<td>600</td>
<td>5</td>
<td>60,000</td>
<td>1214</td>
</tr>
<tr>
<td>100</td>
<td>800</td>
<td>5</td>
<td>80,000</td>
<td>1270</td>
</tr>
</tbody>
</table>
3.2.2. LAN Connection at 100.0 Mbps

3.2.2.1. Shopping Cart Page

[http://localhost:2416/CDShop/ShoppingCart.aspx]

The response time for different combinations of users and requests is noted.

(i) In the following test, the number of loop-count is kept constant at 20,000 and the users are increased at intervals. The maximum number of users is limited to 1000 due to the hardware capabilities. A graph is plot for number of requests (X-axis) versus the response time (Y-axis).

<table>
<thead>
<tr>
<th>Users</th>
<th>Loop Count</th>
<th>Ramp-up period (sec)</th>
<th>Average Response Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>20000</td>
<td>5</td>
<td>2200</td>
</tr>
<tr>
<td>600</td>
<td>20000</td>
<td>5</td>
<td>7328</td>
</tr>
<tr>
<td>800</td>
<td>20000</td>
<td>5</td>
<td>11900</td>
</tr>
<tr>
<td>1000</td>
<td>20000</td>
<td>5</td>
<td>15200</td>
</tr>
</tbody>
</table>
(i) In the following test, the number of users is kept constant and the number of requests is increased and a graph is plot for number of requests (X-axis) versus the response time (Y-axis).

<table>
<thead>
<tr>
<th>Users</th>
<th>Loop Count</th>
<th>Ramp-up period (sec)</th>
<th>Number of Requests</th>
<th>Average Response Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>200</td>
<td>5</td>
<td>20,000</td>
<td>936</td>
</tr>
<tr>
<td>100</td>
<td>400</td>
<td>5</td>
<td>40,000</td>
<td>965</td>
</tr>
<tr>
<td>100</td>
<td>600</td>
<td>5</td>
<td>60,000</td>
<td>976</td>
</tr>
<tr>
<td>100</td>
<td>800</td>
<td>5</td>
<td>80,000</td>
<td>994</td>
</tr>
<tr>
<td>100</td>
<td>1000</td>
<td>5</td>
<td>1,00,000</td>
<td>995</td>
</tr>
<tr>
<td>100</td>
<td>1200</td>
<td>5</td>
<td>1,20,000</td>
<td>1001</td>
</tr>
<tr>
<td>100</td>
<td>1600</td>
<td>5</td>
<td>1,60,000</td>
<td>1014</td>
</tr>
<tr>
<td>100</td>
<td>2000</td>
<td>5</td>
<td>2,00,000</td>
<td>1085</td>
</tr>
</tbody>
</table>

The response time for different combinations of users and requests is noted.

(i) In the following test, the number of loop-count is kept constant at 20,000 and the users are increased at intervals. The maximum number of users is limited to 1000 due to the hardware capabilities. A graph is plot for number of requests (X-axis) versus the response time (Y-axis).

<table>
<thead>
<tr>
<th>Users Count</th>
<th>Loop Count</th>
<th>Ramp-up period (sec)</th>
<th>Average Response Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>20000</td>
<td>5</td>
<td>7400</td>
</tr>
<tr>
<td>600</td>
<td>20000</td>
<td>5</td>
<td>21700</td>
</tr>
<tr>
<td>800</td>
<td>20000</td>
<td>5</td>
<td>28600</td>
</tr>
<tr>
<td>1000</td>
<td>20000</td>
<td>5</td>
<td>35390</td>
</tr>
</tbody>
</table>
(ii) In the following test, the number of users is kept constant and the number of requests is increased and a graph is plot for number of requests (X-axis) versus the response time (Y-axis).

<table>
<thead>
<tr>
<th>Users</th>
<th>Loop Count</th>
<th>Ramp-up period (sec)</th>
<th>Number of Requests</th>
<th>Average Response Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>100</td>
<td>5</td>
<td>5000</td>
<td>1859</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>5</td>
<td>10,000</td>
<td>3149</td>
</tr>
<tr>
<td>100</td>
<td>200</td>
<td>5</td>
<td>20,000</td>
<td>3736</td>
</tr>
<tr>
<td>100</td>
<td>400</td>
<td>5</td>
<td>40,000</td>
<td>3835</td>
</tr>
<tr>
<td>100</td>
<td>600</td>
<td>5</td>
<td>60,000</td>
<td>3910</td>
</tr>
</tbody>
</table>

The Home page of the application is the database extensive page and hence the high response times.
3.1.1.5. Login Web Page [http://localhost:2416/CDShop/Login.aspx]

The response time for different combinations of users and requests is noted.

(i) In the following test, the number of loop-count is kept constant at 20,000 and the users are increased at intervals. The maximum number of users is limited to 1000 due to the hardware capabilities. A graph is plot for number of requests (X-axis) versus the response time (Y-axis).

<table>
<thead>
<tr>
<th>Users</th>
<th>Loop Count</th>
<th>Ramp-up period (sec)</th>
<th>Average Response Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>400</td>
<td>20000</td>
<td>5</td>
<td>5000</td>
</tr>
<tr>
<td>600</td>
<td>20000</td>
<td>5</td>
<td>6323</td>
</tr>
<tr>
<td>800</td>
<td>20000</td>
<td>5</td>
<td>8340</td>
</tr>
<tr>
<td>1000</td>
<td>20000</td>
<td>5</td>
<td>13067</td>
</tr>
</tbody>
</table>
(ii) On this test, the error rate will always be 0%. This is because any user coming to this particular site will be able to access it. The response time to load this page is fast because there are no database requests submitted to the server. The case will be different once a user hits the login button. In the following test, the number of users is kept constant and the number of requests is increased and a graph is plot for number of requests (X-axis) versus the response time (Y-axis).

<table>
<thead>
<tr>
<th>Users Count</th>
<th>Loop Count</th>
<th>Ramp-up period (sec)</th>
<th>Number of Requests</th>
<th>Average Response Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>200</td>
<td>5</td>
<td>20,000</td>
<td>893</td>
</tr>
<tr>
<td>100</td>
<td>400</td>
<td>5</td>
<td>40,000</td>
<td>957</td>
</tr>
<tr>
<td>100</td>
<td>600</td>
<td>5</td>
<td>60,000</td>
<td>1022</td>
</tr>
<tr>
<td>100</td>
<td>800</td>
<td>5</td>
<td>80,000</td>
<td>1050</td>
</tr>
<tr>
<td>100</td>
<td>1000</td>
<td>5</td>
<td>1,00,000</td>
<td>1078</td>
</tr>
</tbody>
</table>

The graph is plot for number of requests (X-axis) versus the response time (Y-axis) and it shows that response-time’s increases as the number of requests are increased.
Also, the login Webpage can stand up to a maximum of approximately 16900 requests with an average response time of 2917ms.

3.1.2. Comparison of the testing results for one of the pages (Home Page)

3.1.2.1. Peak Load at Wireless (54 Mbps) vs. LAN Connection (100 Mbps)

<table>
<thead>
<tr>
<th>Users</th>
<th>Loop Count</th>
<th>Ramp-up period (sec)</th>
<th>Avg. Response Time (ms) for Wireless</th>
<th>Avg. Response Time (ms) for LAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>20000</td>
<td>5</td>
<td>8354</td>
<td>7400</td>
</tr>
<tr>
<td>600</td>
<td>20000</td>
<td>5</td>
<td>22538</td>
<td>21700</td>
</tr>
<tr>
<td>800</td>
<td>20000</td>
<td>5</td>
<td>29567</td>
<td>28600</td>
</tr>
<tr>
<td>1000</td>
<td>20000</td>
<td>5</td>
<td>38603</td>
<td>35390</td>
</tr>
</tbody>
</table>
Observations

1) Response Time increases linearly with number of users for both Wireless and LAN
2) Max no. of users handled by the system before it becomes saturated = 1000
3) Practical no. of users that the application can handle are upto 200. From the graph above, it is seen that the response time is about 7 sec. In practical, users will be reluctant to use the website with such response time. Hence the range of the no. of users this application can handle with a low response time is upto 200.
4) Response Time is less for LAN due to better bandwidth.

3.1.2.2. Comparison of Response Times of all 3 WebPages at Wireless Connection of 54.0Mbps

<table>
<thead>
<tr>
<th>Avg. Response Time (ms) for Login Page</th>
<th>Avg. Response Time (ms) for Shopping Cart Page</th>
<th>Avg. Response Time (ms) for Home Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1900</td>
<td>2500</td>
<td>8354</td>
</tr>
<tr>
<td>7439</td>
<td>7700</td>
<td>22538</td>
</tr>
<tr>
<td>8500</td>
<td>10800</td>
<td>29567</td>
</tr>
<tr>
<td>13000</td>
<td>15400</td>
<td>38603</td>
</tr>
</tbody>
</table>
Note
1) Loop-Count constant at 20,000
2) Ramp-up period of 5 sec
3) Users – 200, 600, 800, 1000

Observations
1) Response Time increases more for Home Page as compared to Login and Shopping Cart Page
2) Lowest Response Time for Login Page as no database requests are submitted by the user
3) Moderate Response Time for Shopping Cart page because there are more computations
4) Response Time for Shopping Cart Page is approx. 28% more on an average than for Login Page
5) Response Time for Home Page is approx. 246% more on an average than for Login Page

Factors affecting Response Time
1) Extensive Data access – numerous read requests handled and images retrieval
2) Varying Network Bandwidth
3) Limited System Hardware Resources (CPU, RAM, Disks) and Configuration
4) JMeter Tests and Server running on the same machine

3.1.2.3. Comparison of Constant Users vs. Constant Loop-Count of Home Page for Wireless Connection

(i) Users Constant at 200 and Loop-Count increased up to 20000
Observations

1) Response Time increases rapidly with number of users but not very much when the users are kept constant and only loop-count is increased.

This is because, if the number of users is kept constant and only the loop count is increased, the number of requests handled by the server per second remains constant for every execution of the loop-count and for every increase in the loop count. Hence the response time will not increase drastically in this case. Whereas, if the users are increased and loop count is kept constant, the requests handled by the server per second increases with increasing users and hence the longer response time.

3.1.3. Wireless Connection at 54.0 Mbps for Sustained Load
3.1.3.1. Login Web Page [http://localhost:2416/CDShop/Login.aspx]

<table>
<thead>
<tr>
<th>Users</th>
<th>Loop Count</th>
<th>Ramp-up period (sec)</th>
<th>Average Response Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>800</td>
<td>16000</td>
<td>300</td>
<td>10335</td>
</tr>
</tbody>
</table>

3.1.3.2. Shopping Cart Page [http://localhost:2416/CDShop/ShoppingCart.aspx]

<table>
<thead>
<tr>
<th>Users</th>
<th>Loop Count</th>
<th>Ramp-up period (sec)</th>
<th>Average Response Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>800</td>
<td>20000</td>
<td>300</td>
<td>10197</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Users</th>
<th>Loop Count</th>
<th>Ramp-up period (sec)</th>
<th>Average Response Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>800</td>
<td>16000</td>
<td>300</td>
<td>30670</td>
</tr>
</tbody>
</table>
3.1.4. LAN Connection at 100.0 Mbps for Sustained Load


<table>
<thead>
<tr>
<th>Users</th>
<th>Loop Count</th>
<th>Ramp-up period (sec)</th>
<th>Average Response Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>800</td>
<td>15000</td>
<td>300</td>
<td>17271</td>
</tr>
</tbody>
</table>

3.1.4.2. Login Web Page [http://localhost:2416/CDShop/Login.aspx]

<table>
<thead>
<tr>
<th>Users</th>
<th>Loop Count</th>
<th>Ramp-up period (sec)</th>
<th>Average Response Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>800</td>
<td>17000</td>
<td>300</td>
<td>12897</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Users</th>
<th>Loop Count</th>
<th>Ramp-up period (sec)</th>
<th>Average Response Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>700</td>
<td>16000</td>
<td>300</td>
<td>27737</td>
</tr>
</tbody>
</table>
4. Scalability

The Online Music Store System is running on a personal version of IIS. A better scalability can be achieved by upgrading a personal version of IIS to a server version of IIS. The system database server and the IIS server are running on the same machine. An increased performance and scalability can be achieved by using a dedicated database and a web server.
Chapter 9 – User Manual

1. Introduction

This document will explain how to set up and use the Online Music CD Store.

2. Installation and Set-up

2.1. Required Hardware

- A Microsoft SQL Database Server
  - Servers should be equipped with a processor clock speed over 1.5GHz and a memory over 512mb.
- IIS Web Server
  - Servers should be equipped with a processor clock speed over 1.5GHz and a memory over 512mb.

2.2. Required Software

- Microsoft Windows 2000 Professional or Windows XP Professional
- Internet Information Server (IIS). The IIS Web server is included on Windows 2000 Professional and Windows XP Professional installation CD. (It is not automatically installed so it needs to be installed.)
- .NET Framework 2.0 SDK which may be downloaded from this site: http://www.microsoft.com/downloads/details.aspx?familyid=fe6f2099-b7b4-4f47-a244-c96d69c35dec&displaylang=en
- Microsoft Internet Explorer 6.0.2 (or higher) or Mozilla Firefox for clients
- Microsoft SQL Server 2005 which is installed with Microsoft visual studio 2005.
- SQL Server 2005 Express Manager which can be downloaded from this site: http://msdn2.microsoft.com/en-us/express/aa718378.aspx
- Microsoft Visual Studio 2005 for configuring connection string for the database.

2.3. Required Network Configuration

A broadband, having a 10Mbp and above, connection is needed to provide fast linkage to the server, thus granting quality service to the user.
2.4. Software Set-up

- Unpack the software to the World Wide Web root of the IIS server. (It is usually the “C:\inetpub\wwwroot” directory.)
- Open the “Web.config” file and change the values of the <add name> property according to your connection configurations
- Open the SQL-Server Manager Studio Express
- Create a database and label it as “MusicStore”
- Click on the query analyzer
- Paste the contents of the “musicstore.sql” file
- Select the Database Name “MusicStore”
- Copy and paste all the contents of the sql file and press the “F5” button on your keyboard (This will create all tables and stored procedures in the MusicStore database)
- Open the Internet Information Services.
  - Go to Start → Control Panel → Administrative Tools
• Choose the option “Web Sites”

• Right click on “Default Web Site”

• Click on New ➔ Virtual Directory

• Click on the “Next” button
• Enter the Alias Name as “MusicStore” then click on the “Next” button

• Browse the directory and select the folder where you have unpacked the software, then click on the “Next” button

• Click on the “Next” button
• Finally, click on the “Finish” button

• Click on the “MusicStore” folder and select the “Default.aspx”

• Right click on the “Default.aspx” and select the “Browse” option
3. Online Music Store Usage

3.1. User

3.1.1. Manage Account

3.1.1.1. Register

The purpose of this part of the application is to provide user authentication. The user will enter six inputs (username, password, confirm password, emailed, security question and answer) through the keyboard. The system then checks both username if it is already in use by other user, if yes, the user will be prompted to re-enter another username. If the username is valid, the system will check the password and confirm password fields. If they do not match appropriate error message will be displayed. After having filled all the required fields correctly and registered as a valid user, user will be directed to the login page.
3.1.1.2. Login

The purpose of this part of the application is to provide user authentication. The user will enter two inputs (username and password) through the keyboard. The system then checks both username and password to see if they are valid. If the username or password is not valid, the appropriate error message will be displayed and the user needs to re-enter the username and password. If the user inputs are valid, the user will be directed to the requested secure web page and the default Web page will be displayed.
If the user is logged in, the logout link will be displayed on the user menu located on the left side of the screen. Users can logout anytime by clicking on the “Logout” link.

3.1.1.3. Manage Profile

The purpose of this part of the application is to enable the user to edit his personal information like billing address, shipping address and credit card information. The user will enter valid information in all the required fields before placing the order.
3.1.2. Search

Users can search for available items by clicking on the “Search” button found on the top of the welcome screen. This will redirect the user to “search.aspx” webpage. There are two ways for the user to search. First, the user can search the catalog by typing the text in the textbox and checking the “Search for all words option”. This will display the items whose name matches with the text typed in by the user. Secondly; the user can also search for a particular album
by entering any of the following options: title of the song, artist, style, format and the price range and then click the “Search” button. This will display the matched albums and then the user can select each of those to see their details.

3.1.3. Get Song Recommendations

Users can also get recommendations for the songs of their choice. The user will enter the song name, the artist and the style of the song for which he/she wants to find recommendations. Then the user will click “Get Similar Songs” button. This will give the 10 most similar songs according to the options entered by the user. The user will then select any three of those songs at one time and click the “Get Recommendations” button. This will give their 5 most recommended songs.
3.1.4. Browse

Users can also view information about the different genres and their corresponding categories of songs. This can be done by clicking on the “Choose a Genre” and “Choose a Category” links located on the left side of the screen.
Users can also see the details of each product individually. This can be done by clicking on the item image or the product name. This will redirect the users to the product detail page.

3.1.5. Place Order

3.1.5.1. Add to Cart

The purpose of this part of the application is to enable the user to add the items to the shopping cart. This can be done by clicking the ‘Add to Cart’ button below the product details. The items added by the user will get added one by one to the Cart Summary block located on the left side of the screen.
3.1.5.2. View Cart Details

The user can see the details of his cart by clicking on the “View Details” button from the Cart Summary block located on the left side of the screen. This will display the details such as the item name, its unit price, quantity of the item, the subtotal cost of each item based on its quantity and also the total cost of all the items in the cart.
3.1.5.3. Edit Item Quantity

The user can also increase or decrease the quantity of the items in the shopping cart by writing the number of quantity in the text box across the respective item and clicking the “Update” button. The user can also remove a particular item from the shopping cart by clicking the “Delete” button on the side of the item.

3.1.5.4. Checkout and Place order

If the user has finished adding the items in his shopping cart, he can checkout the shopping cart by clicking “Proceed to Checkout” button located at the bottom of the cart. This will redirect the user to the page where he can confirm his billing address, shipping address and select the shipping type. If all the information is valid, the user can place an order by clicking the “Place Order” button at the bottom of the screen.
These are the products in your shopping cart:

<table>
<thead>
<tr>
<th>Product Name</th>
<th>Price</th>
<th>Quantity</th>
<th>Subtotal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bach And Beyond</td>
<td>22.00</td>
<td>1</td>
<td>22.00</td>
</tr>
<tr>
<td>Wagner: Der Fliegende Holländer</td>
<td>26.00</td>
<td>1</td>
<td>26.00</td>
</tr>
<tr>
<td>The Indispensable</td>
<td>21.99</td>
<td>1</td>
<td>21.99</td>
</tr>
<tr>
<td>Inspired By Bach</td>
<td>22.99</td>
<td>1</td>
<td>22.99</td>
</tr>
<tr>
<td>Giselle</td>
<td>21.99</td>
<td>1</td>
<td>21.99</td>
</tr>
</tbody>
</table>

Total amount: $108.97

Billing Address
Name: Reshma Sawant
Address: 1011 Laramee Street Apt # 1
City: Manhattan
State: Kansas
Zip / Postal Code: 66502
Country: United States

Shipping Address
Name: Sarang Mairal
Address: 551 Sunset Avenue Apt #11
City: Manhattan
State: Kansas
Zip / Postal Code: 66502
Country: US/Canada
Contact no: 765-313-6115
Email: reshma@ksw.edu
Credit Card: XXXX-XXXX-XXXX-3456

Please confirm that the above details are correct before proceeding.

Shipping type: Next Day Delivery (No)
3.2. Administrator

3.2.1. Login

The purpose of this part of the application is to provide administrator authentication. The administrator will enter two inputs (username and password) through the keyboard. The system then checks both username and password to see if they are valid. If the username or password is not valid, the appropriate error message will be displayed and the administrator needs to re-enter the username and password. If the administrator inputs are valid, the user will be directed to the requested secure web page and the index Web page will be displayed.

If the administrator is logged in, the logout link will be displayed on the administrator menu located on the left side of the screen. Administrators can logout anytime by clicking on the “Logout” link.
3.2.2. Add and Delete Genre

The administrator can create a new genre or delete an existing genre by clicking the “Catalog Admin” link found at the administrator menu located on the left side of the screen. To create a new genre, administrator will enter the name and description in the provided text box and click the “Create Genre” button. The information will be updated accordingly and the new genre will be saved on to the database. If a new genre is created successfully, an appropriate message will be displayed. The administrator can also delete an existing genre by clicking the “Delete” button located on the right side of each corresponding genre. If a genre is deleted successfully, an appropriate message will be displayed.
3.2.3. **Add and Remove Product**

The administrator can create a new product or delete an existing product by clicking the “Catalog Admin” link found at the administrator menu located on the left side of the screen. To create a new product, administrator will enter the name, description, price and the image in the provided text box and click the “Create Product” button. The information will be updated accordingly and the new product will be saved on to the database. If a new product is created successfully, an appropriate message will be displayed. The administrator can also select a product to view its details and can also remove the product from a category/genre or move it.
within the existing categories. Administrator can also edit the details of products by clicking the “Edit” button located on the right side of each product.

3.2.4. **Edit Catalog Details**

The Administrator can also edit details for existing genres and products such as genre name, genre description, product name, product description, price etc. This can be done by clicking the “Edit” button located at the right of each product and genre.
3.2.5. Manage Orders

The purpose of this part of the application is to enable the administrator to manage the details of the orders created by the users. The administrator can view the most recent orders created, orders created between certain period by entering valid dates, the unverified & uncanceled orders, verified & uncompleted orders. If an invalid number or date is entered or any of the text boxes are left blank, an appropriate error message in displayed and administrator has to reenter the value.

Administrator can also select a particular order to view the items contained by that order and to mark it as verified, canceled or completed.
and to update its details such as Customer name, Shipping address, date created, date shipped and customer email.
3.2.6. Manage Shopping Carts

The administrator can also view how many old shopping cart entries exists and can delete them if they haven’t been updated for a certain amount of time.
Chapter 10 – Project Evaluation

1. Introduction

   This document presents the summary of experiences gained during the entire life-cycle of the developer’s MSE project.

2. Problems Encountered

   This section describes the problems and difficulties encountered during the project.

2.1. Solution domain research and learning

   One of the first and basic problems was identifying which technology to use for developing the project. There were two options to choose from: J2EE or Microsoft .NET. Both tools had their advantages and disadvantages. I chose the latter option because of its ease of use plus it had lesser configurations to perform as compared to J2EE.

2.2. Learning ASP.NET and C# Language

   It took quite some time for me to get used with the programming languages. Although, C# was very much similar to Java and ASP.NET with Visual Basic, still I found myself encountering some challenges in terms of finding a good book that covers in-depth examples. My basic knowledge of connecting with databases to send the information was not enough. Mainly dealing with the datasets and adapters concept in C#.NET was a major learning curve. Also working with the namespaces/assemblies in the .NET framework initially took major time. Most examples found in the Internet are very basic ones and so I ended up spending a lot of my time searching for good examples with good codes.
2.3. Security Issues

My computer setting was set to having automatic security updates. It has been a notion for me to think that it is safe to do so until I have encountered some issues with the security policy. Some functionalities of the application, that used to work, were no longer functional. I spent a lot of time trying to pin down where the actual problem is originating, but still I was not lucky. My last option was to redo that part of the solution project. My time got wasted in redoing the coding for that particular function and still I was not able to figure out what the real problem was.

The database connection constrains was the major information not to be visible to the users. For this I have used a single connection with the database which is stored in web.config file of the project which helped in reusability of the connection string and to maintain the security.

3. Source Lines of Code

The initial estimate was 3200 SLOC in Phase II based on the current progress of the developer driven from the prototype made in Phase II. The actual SLOC is 3795. There is a difference of 595 SLOC. Reasons that have caused the difference between the estimation and actual number were: inexperience of the developer and unfamiliarity with the programming language and database used, and the auto-generated code by the IDE used.

Below is the estimate of the C# LOC counted by LocMeter tool:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Count</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Files</td>
<td>48</td>
<td>Source Files</td>
</tr>
<tr>
<td>Directories</td>
<td>10</td>
<td>Directories</td>
</tr>
<tr>
<td>LOC</td>
<td>3625</td>
<td>Lines of Code</td>
</tr>
<tr>
<td>BLOC</td>
<td>570</td>
<td>Blank Lines of Code</td>
</tr>
<tr>
<td>SLOC_P</td>
<td>3376</td>
<td>Physical Executable Lines of Code</td>
</tr>
<tr>
<td>SLOC_L</td>
<td>2757</td>
<td>Logical Executable Lines of Code</td>
</tr>
<tr>
<td>MNG</td>
<td>248</td>
<td>McCabe VB Complexity</td>
</tr>
<tr>
<td>C&amp;CLOC</td>
<td>2</td>
<td>Code and Comment Lines of Code</td>
</tr>
<tr>
<td>LOC</td>
<td>1079</td>
<td>Comment Only Lines of Code</td>
</tr>
<tr>
<td>CWORD</td>
<td>2432</td>
<td>Commentary Words</td>
</tr>
<tr>
<td>HLOC</td>
<td>0</td>
<td>Header Comment Lines of Code</td>
</tr>
<tr>
<td>HWORD</td>
<td>0</td>
<td>Header Commentary Words</td>
</tr>
</tbody>
</table>
As observed from the above table, the executable physical SLOC for C# code is 2757.

The following is the breakdown of the project’s SLOC:

- Total C# code - 2757
  - Handwritten C# Code – 2057 (approx)
  - Auto-generated C# code – 700 (approx)
- SQL Code - 540
- XML Code - 86
- CSS Code - 412

### 4. Project Duration

The initial estimated time from the COCOMO model analysis of the project is 4.5 staff months. But after completing Phase II (Nov.2007), I estimated the required days to be 78 for Coding/Debugging, Testing and Documentation. 78 Days after Nov. 2007 comes to approx. mid Feb. 2007. The actual time spent on the entire project is about 7.5 months which is approximately 1 month more than the estimate. It took me that amount of time to complete the project because I consider myself a novice programmer in the technologies I have chosen with some major functionalities being implemented.

The following table shows the entire duration of the project:

<table>
<thead>
<tr>
<th></th>
<th>Start Time</th>
<th>Finish Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase I</td>
<td>Aug 1, 2007</td>
<td>Sept 20, 2007</td>
</tr>
<tr>
<td>Phase II</td>
<td>Oct 1, 2007</td>
<td>Nov 27, 2007</td>
</tr>
<tr>
<td>Phase III</td>
<td>Dec 5, 2007</td>
<td>March 10, 2008</td>
</tr>
</tbody>
</table>

Table 19. Project Duration

The following table shows the breakdown of activities in each phase:

<table>
<thead>
<tr>
<th>Phases</th>
<th>Phase I Hours</th>
<th>Phase II Hours</th>
<th>Phase III Hours</th>
<th>Total Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Research</td>
<td>21</td>
<td>14</td>
<td>3.5</td>
<td>38.5</td>
</tr>
</tbody>
</table>
The following pie chart shows the breakdown on how much time was spent on each phase:

![Breakdown for Each Phase]

The following pie chart shows the breakdown on how much time was spent on Phase I:

![Phase I Breakdown]
The following pie chart shows the breakdown on how much time was spent on Phase II:

![Phase II Breakdown](image)

The following pie chart shows the breakdown on how much time was spent on Phase III:

![Phase III Breakdown](image)
5. Lessons Learned

This project has been the most rewarding experience I have had in my entire master’s degree program. It presented me a good exposure on what is it like to work on real-world project. Mostly I worked in teams when doing large projects and I never thought I would finish a “large-scale” project such as this by myself. I have learned how to undertake certain tasks under pressure. Going through the entire life cycle process of software development made learn how to get organized in everything. The set of documents that I have produced have given me an insight into the necessity and usefulness of rigorous documentation. Also, revising with the Class Diagrams, working more closely with UML with respect to Web related project was a great opportunity and it will help me in future projects.
REFERENCES


5. John Kauffman and Bradley Millington. Beginning ASP.NET 2.0 and Databases (Wrox Beginning Guides).


Appendix A. Srunokshi Neelakantan’s Formal Inspection Checklist

1. **Purpose**

   The purpose of this document is to provide a checklist for the technical inspectors of the Online Music CD Store Website. The goal of the technical inspection process is to aid the developer in checking for correctness and consistency with the architectural design, which ensures the quality of the software design.

2. **Items to be Inspected**

   - UML Diagrams
     - Class diagrams
     - Sequence diagrams
   - Formal Specification
   - USE Model

3. **Formal Technical Inspectors**

   - Rahul Deshmukh
   - Srunokshi Neelakantan

4. **Formal Technical Inspection Checklist**

<table>
<thead>
<tr>
<th>Item to be Inspected</th>
<th>Pass/Fail/Partial</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>10. The documentation follows the MSE portfolio standard which is described at <a href="http://mse.cis.ksu.edu/oncampus/mse-portfolio.htm">http://mse.cis.ksu.edu/oncampus/mse-portfolio.htm</a></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11. All the classes in the USE model are represented in the class diagram.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12. The multiplicities in the USE model have been depicted in the class diagram.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13. All the symbols used in the class diagrams are according to the UML standards.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>14. All the classes in the class diagram are clear as to what they represent in the architecture design document.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15. Class Diagram matches the code parameters.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16. The symbols used in the sequence diagram correspond to UML standards.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17. Sequence diagram matches class diagram.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18. All the requirements in the Software Requirements Specification have been covered in the Architecture Design Document.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Appendix B. Inspection Letter

1. Srunokshi Neelakantan’s Inspection Letter

Dear Dr. Andresen,

I have completed the Formal Technical Inspection for Reshma Sawant MSE project. I reviewed her Architecture Document including the USE Model. I found everything satisfactory. All of the items in the checklist passed.

Regards,
Srunokshi Neelakantan.