Designing and Evaluating a Recommender System within the Book Domain

by

Monira Essa AL-Oud

25 August 2008

Scheme: MSc E-Commerce Technology

Supervisor: Dr. Maria Fasli

Department of Computing and Electronic System

University of Essex

Abstract

Today the World Wide Web provides users with a vast array of information, and commercial activity on the Web has increased to the point where hundreds of new companies are adding web pages daily. This has led to the problem of information overload. Recommender systems have been developed to overcome this problem by providing recommendations that help individual users identify content of interest by using the opinions of a community of users and/or the user's preferences.

The aim of this thesis was to design and evaluate different approaches for producing personalised recommendations within the book domain. To achieve this goal, the project first investigated existing recommender systems and profiling techniques. The next step was to build users' profiles by monitoring users' behaviour, and develop three different approaches for producing recommendations. Finally, an evaluation of the system recommendations' accuracy was done, by first conducting live user experiments and then performing offline analysis to measure the recommendations' accuracy using appropriate methods for testing.

The system evaluation results show that the accuracy of the system recommendations is very good and that a recommender system based on the combination of content-based and collaborative filtering approaches provides more accurate recommendations for the book domain.

Acknowledgment

All praises and thanks are addressed to Allah for giving me the strength to complete my MSc program successfully.

I express my tremendous appreciation to my great father and mother who have always supported and encouraged me to work hard. I would like to express my deep thanks to my husband and son, who have always understood my ambition to achieve this goal and have support my efforts. I extended my gratitude to my brothers Mohammad and Saud and my sisters Reem and Arwa for their continuous support.

My sincere acknowledgments go to my advisor, Dr. Maria Fasli, for her guidance, immeasurable support and constant feedback in conducting and completing this project.

Finally, I would like to express my gratitude to all the participants of this project experiment, whose punctual participation led to the success of this project.

Table of Contents

Table of Figures

Table of Tables

1 Introduction

1.1 Project type

This is an experimental project which first, designs…. And second evaluates different approaches for offering recommendations to readers regarding books they may wish to purchase, as part of an online bookshop website.

1.2 Motivation

Today the World Wide Web has provided access to a vast array of information through the web pages, as a result of the Internet growth. Also, commercial activity on the Web has increased to the point where hundreds of new companies are adding Web pages daily. With this increase in the information sources, a problem of information overload occurs, in which the users are trying to deal with an excess of information that is not useful to them as they try to make sensible decisions (Losee, 1989). As a response to this problem, a range of tools to help with retrieving, searching and filtering have been developed.

The tool most widely used to alleviate the problem of information overload is the search engine. The benefits for the users from search engine technology have decreased as the number of web pages has grown. In addition, the user must first consider the large number of search tools available and decide which one to access. Then the user must interact with each one individually because search engines are typically not personalised to individual users or their prevailing context. Users usually make a choice on the basis of their personal experience or other people's experience. Based on these facts, recommender systems have been developed to provide recommendations that help individual users identify content of interest by using the opinions of a community of users and/or the user's preferences.

Today various recommendation systems play an important role in supporting commercial websites to help users find items that they know they would like to purchase, as well as discover new items about which they had been unaware. The ability to persuade the consumers to buy a suitable item is a significant goal for any recommender system in an e-commerce environment. However, for any recommender system to be successful, the consumer must trust and accept the system's recommendations. This is done with a clear explanation from the system, presented in a way that is in keeping with the consumer's

preferences. A good recommender system can significantly contribute to achieving the consumer's acceptance of the system recommendations.

1.3 Project Scope

The project will design and evaluate a real online book shop recommender system that is based on both content-based and collaborative filtering technologies.

Typically, content-based recommendations require machine learning techniques in order to find patterns in the items users prefer (Middleton, 2003). The content-based technology will be related to real users' experiences. It will build users' profiles to observe their behaviour. Moreover, the system will generate recommendations by analyzing the similarity between the contents of the books that are in the user's profile and those that the user had not evaluated.

In contrast to content-based recommendations, collaborative filtering recommendations are based on the opinions of a community of similar users. The collaborative filtering recommendations will be created by asking the users to rate books; this will allow the system to recommend new books that similar users have already rated highly. Collaborative filtering requires a statistical technique to compute the similarity between similar users' preferences (Middleton, 2003).

In addition, each of content-based and collaborative filtering technologies used in the project will embed a specific algorithm to compute the similarity in order to generate recommendations. In addition, I will experiment with techniques to combine content-based and collaborative filtering technologies to improve the system's recommendations.

The different approaches for computing recommendations will be tested with real end-users to determine the accuracy of the recommendations. Testing includes user experiments, in which the users evaluate a list of recommend books while the system receives user feedback. Evaluation will be done using offline analysis and statistical accuracy metrics to evaluate the accuracy of a recommender system.

1.4 Project Aim and Objectives

1.4.1 Project Aim

This project aims to design and evaluate different approaches for computing recommendations within the book domain to provide personalised recommendations to the users.

1.4.2 Project Objectives

The project has the following four objectives:

1. Investigate and evaluate existing recommender system and profiling techniques.
2. Build a user's profile for a recommender system by monitoring dynamic user behaviours. The user profile must adapt to the changing in user's interest.
3. Develop a recommender system using different approaches for computing recommendations.
4. Evaluate the accuracy of the system recommendations using an appropriate methodology.

The research will would provide answers to the following questions:

- Is the recommender system an efficient approach to solve the problem of information overload in e-commerce web sites?
- Which is the most accurate recommendation technology for the book domain?
- Does the system provide accurate recommendations to the users?

1.5 Methodology

1.5.1 Implementation methodology

The project will be developed through the adoption of an incremental approach to software development. The incremental approach is the process of breaking down the delivery of the complete solution into manageable increments that can be developed and stored securely (Sommerville, 2006). An incremental delivery process has the benefit of earlier knowledge of problems; this allows adaptation of the plan and good maintenance of the system. Also, the

project manager will benefit at each delivery milestone by knowing the status of that delivery, instead of compounding delays for the whole system.

A general process model for incremental development (Sommerville, 2006) is illustrated in **Figure 1.1**.

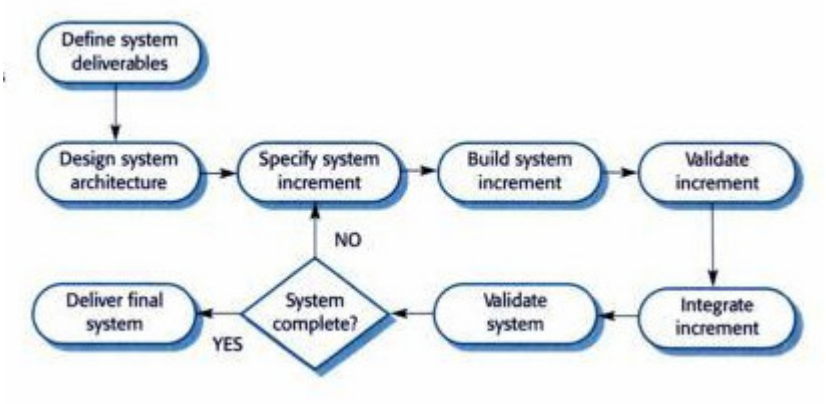


Figure 1.1: An Iterative Development Process (Sommerville, 2006, p. 394)

1.5.2 Experiment Methodology

One of the most important key decisions regarding the evaluation of a recommender system algorithm is the choice of whether to carry out the evaluation offline on an existing data set or whether to conduct a live user experiment. The advantage that off-line evaluation has over live user experiments is that it is quick and economical to conduct large evaluations on different algorithms at several datasets. Once a dataset is available, conducting the experiment only requires running the algorithm on the appropriate subset of the data (Herlocker et al, 2001). In contrast, a live user experiment is evaluated on users of a running recommender system which can evaluate the users' satisfaction about the system. The on-line evaluation needs a fully developed system and a community of users; while for the off-line analysis these are not required (Hayes et al., 2002).

I have chosen to conduct live user experiments to evaluate the recommender system algorithms because there is no efficient existing data set for the book domain, and because I want to evaluate the algorithms under different conditions.

The project experiment will consists of developing an off-line application which will be used by a community of users. The application will contain three different approaches for

computing recommendations, in order to evaluate the relative advantages of the approaches. These three approaches will run under the same conditions and have access to the same resources. The dataset for the proposed recommender system will be collected in live user experiments. The user will evaluate a list of recommended books while the system receives feedback. Then, statistical accuracy metrics will apply to evaluate the accuracy of each algorithm. In addition, the project will conduct off-line analysis to evaluate the sensitivity of different parameters in the recommender system algorithms to determine the sensitivity of these parameters.

1.6 Thesis Structure

Chapter 1: Project Description

This thesis begins by introducing, in Chapter 1, the project motivation and scope, detailing what the system hopes to achieve and what techniques will be used for producing recommendations. The project's aim and objectives are then discussed, along with the development methodology that will be used.

Chapter 2: Background

Chapter 2 presents the background of the project. Introduces the information overloads problem and the ways that recommender systems deal with it is. This chapter also describes filtering and profiling techniques that are used by recommender systems. Finally, the second chapter outlines the features of some currently existing systems that fulfil similar functions.

Chapter 3: Requirements

The next chapter gives a brief overview of the system requirements, and then describes the functional and non-functional requirements for the proposed system.

Chapter 4: Design

Chapter 4 details the design of the system increments. Also, it discusses in detail the design of the recommender system module along with the proposed algorithms.

Chapter 5: Implementation

Chapter 5 describes the steps taken to implement the system functionality. Example code is supplied to aid in the reader's understanding.

Chapter 6: Testing and Evaluation

This chapter deals with the testing and evaluating of the system with respect to the requirements.

Chapter 7: Conclusion

The final chapter of the document analyses the achievement of the project's objectives and outlines the management of the project. It also presents conclusions about of the project and suggests possible future work.

2 Background

This chapter will introduce the information overloads problem and the ways that recommender systems deal with it is. Also, it will describe filtering and profiling techniques that are used by recommender systems. The features of some currently existing systems that fulfil similar functions are also explained.

2.1 The Problem of Information Overload

The Internet is growing at least 10 percent per month and the content of the Web grows by an estimated 170,000 pages daily (Turban et al., 2000). As a result the World Wide Web provides the users with a vast array of information that makes it difficult for the users to decide exactly what they want. This phenomenon, of users trying to deal with more information that does not enable them to make sensible decisions, is called information overload (Losee, 1989).

To solve the information overload problem, a range of tools to help with retrieving, searching, and filtering have been developed. The most widely used tool that assists with the problem of information overload is the search engine. Search engines are effective at filtering pages, but people find articulating what they want as a search query difficult and time consuming. The benefits for the users from search engine technology have been decreased over time by the rapid increasing of the web pages.

2.2 Recommender Systems

Recommender systems have been developed to overcome the above mentioned limitations of searching through the massive volume of information available. Recommender systems, in comparison with other filtering tools, require less experience on the part of the user and less effort to specify their interests when querying and operating the system (Resnick and Varian, 1997).

Recommender systems intend to provide users with suggestions of items that they may be interested in, based upon their past preferences, history of purchase, or demographic information, as well as the environment of possible items. In addition, a recommender system helps the site adapt itself and provide individual personalisation for each consumer; this increases the sales for the commercial site (for more details see **Appendix A**).

2.3 Providing Recommendations

Different forms for providing recommendations have been developed; they can be classified into the following forms: attribute-based recommendations, item-to-item correlation, people-to-people correlation and non-personalised recommendations (Konstan et al., 2001). For more detailed descriptions, please see **Appendix B**.

2.4 Recommendation Technologies

Recommendations systems rely on different technologies for computing recommendations. The most important approaches are content-based filtering and collaborative filtering. Content-based filtering displays users as individuals, while recommender systems employing the collaborative filtering approach display the user as a part of a group (Fasli, 2006). In addition, an advanced recommender system that combines content-based and collaborative filtering to avoid the limitations of each approach, is called a hybrid approach.

2.4.1 Content-Based Filtering Approach

The content-based filtering approach identifies the similarity between a user and the new items using the content of the previously evaluated items in the user profile. In addition, each item in a user profile is characterized by a set of attributes which is constructed by extracting a set of features from an item. Such a profile is used to determine if the new item is similar to the item that a user has preferred in the past. For instance, the Newsweeder is a netnews-filtering system that suggests news articles to the user based on the user's profile (Lang, 1995). Most content-based approaches are performed on textual documents, such as web pages and articles. The textual document can be easily broken down into individual words, unlike video and physical resources, which required sophisticated analysis.

Content-based filtering has some shortcomings in recommending items. A user's selection is based on the subjective attributes (such as the quality) of the item (Goldberg et al., 1992); in contrast, content based approaches are based on objective attributes (such as the description of an item) about the items. Also, some items the users may be interested in cannot be recommended to them because content-based methods compare new items with the items previously seen by the user, while the user's interests may be beyond the scope of the previously seen items. Finally, multimedia technology such as sound, video or physical items cannot be analysed automatically for relevant attribute information, due to limitations of resources (Jennings et al., 2005).

2.4.2 Collaborative Filtering Approach

Collaborative filtering recommendations are based on the opinions of a community of similar users. The basic idea is that users recommend items to one another. Collaborative filtering makes this possible by asking the users to rate items, which allows the system to recommend new items that similar users have rated highly. For instance, MovieLens is a movie recommender system that uses collaborative filtering to help people find movies they will like in the huge stream of available movies. Collaborative filtering works well for multimedia technology such as music and movies. However, it also has some limitations:

New user problem: A new user starts off with a profile of interests from scratch. The system needs to know the user preferences in different items to generate accurate recommendations.

Cold start problem: New items cannot be recommended until more information is obtained when another user either rates an item or provides feedback on the item (Fasli, 2006). As a result, the recommendations generated by the system will not recommend items similar enough to the users' interests.

Scalability: A collaborative filtering algorithm should address the scalability issue as the number of users increase and their collective profile size becomes large (Fasli, 2006).

The schematic diagram of the collaborative filtering process is showed in **Figure 2.1**. As you can see from the figure, there is a list of users denoted by U= {u_1, u2,…,u_m} and a list of items I={i_1,i_2,….,i_n}. Each user has a list of items. The collaborative filtering algorithm will generate recommendations, a list of N items that the active user will mostly like, according to the active user. Also, the process will output a prediction, which is the result prediction on item j for the active user (Sarwar et al., 2001).

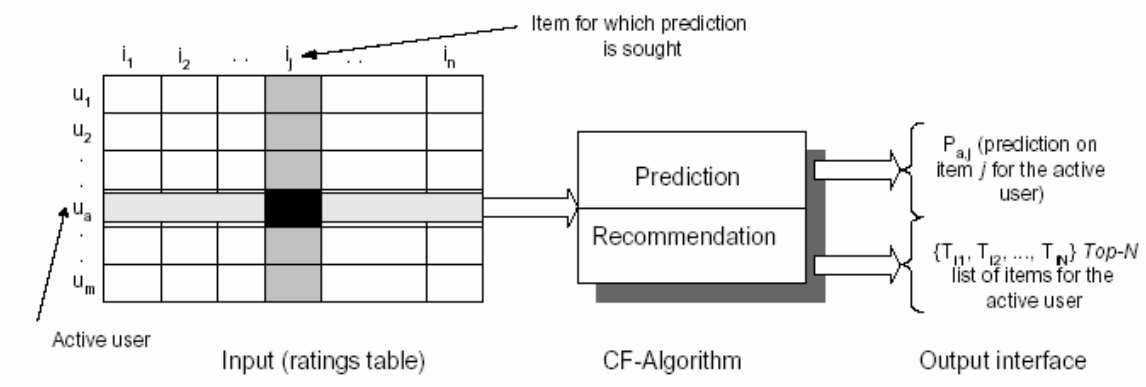


Figure 2.1: The Collaborative Filtering Process (Sarwar et al, 2001, p. 4)

2.4.3 Hybrid Approach

Hybrid approach introduced to combine the advantages of both content-based and collaborative filtering techniques help to overcome their limitations. These use the strength of one set of filtering techniques to overcome the limitation of the other. The hybrid filtering approach is also called "collaborative via content" because content-based profiles are also taken when identifying the similarities among users for collaborative recommendations (Pazzani, 1999).

2.5 Collaborative Filtering Approaches

Researchers have divided a number of collaborative filtering algorithms into two main categories: memory-based (user-based) and model-based (item-based) algorithms (Sarwar et al, 2001).

2.5.1 User-Based Collaborative Filtering

User-based algorithm is based on the fact that each user belongs to a larger group of similarly behaving individuals. It uses statistical techniques to find a set of users with similar interests, known as neighbours, in the entire user-item database, to generate a list of recommendation for the active user (Middleton, 2003).

Different measures of similarity that are based on neighbourhood algorithms are used to compute the similarity between the active user and other users in the database, such as the Pearson correlation coefficient and Mean squared differences algorithms (Breese et al., 1998). Moreover, to predict the rating of an item given by the active user, the ratings from the most similar users for the item are averaged and weighted by their similarities to the active user. The Pearson Correlation reflects the degree of linear relationship between two variables and ranges from +1 to -1. A positive correlation means that the two users have very similar tastes, while a negative correlation indicates that the users have dissimilar tastes (Fasli, 2006). The Pearson Correlation Coefficient method defines the similarity between two users by:

$$r_{xy} = \frac{\sum_{i=1}^{N} (U_{xi} - \bar{U}_x) * (U_{yi} - \bar{U}_y)}{\sqrt{\sum_{i=1}^{N} (U_{xi} - \bar{U}_x)^2 * \sum_{i=1}^{N} (U_{yi} - \bar{U}_y)^2}}$$

r_{xy}	pearson-r correlation between user x and y
N	number of ratings
U_{xi}	i^{th} rating for user x
$\bar{U}_x$	mean rating for user x

Figure 2.2: Pearson Correlation Coefficient Algorithm (Middleton, 2003, p. 34)

Mean Squared Difference (MSD) is the average squared difference between the active user's ratings and the other users' ratings on similar items. This similarity algorithm is used in the Ringo music recommender (Shardanand and Maes, 1995). $msd_{a,u}$ is the mean squared difference between the active user *a* and user *u*. If both users have similar ratings for all items, $msd_{a,u}$ will be 0. Otherwise, $msd_{a,u}$ will be 1.

$$msd_{a,u} = \frac{\sum_{i=1}^{m} (r_{a,i} - r_{u,i})^2}{m}$$

However, a user-based algorithm has some drawbacks related to scalability and real-time performance (Karypis, 2001).

2.5.2 Item-Based Collaborative Filtering

The item-based algorithms are developed to overcome the scalability on user-based recommendations. Unlike a user-based approach, the item-based approach identifies the set of items that are similar or related to the item that the active user has evaluated. After that, it computes the similarity between items and then selects the most similar items to the target item within the set of items that the user has rated (Sarwar et al., 2001).

2.6 Profile Representation

Content-based technology works by building users' profiles in a recommender system to observe their dynamic behaviour. This requires machine learning techniques in order to find patterns in the items users prefer (Middleton, 2003). The content-based and collaborative filtering techniques are applied on the user profiles to identify the relevant items and users, and then produce recommendations.

In order to build a user profiles and provide recommendations, the system must collect data about the user, explicitly, implicitly or both, to provide appropriate recommendations to the user. With explicit profiling, the users directly specify their interests by rating items; this is called ratings-based representation. In implicit profiling, the user's behaviour is tracked automatically by the system; this is called content-based representation (Middleton, 2003).

2.6.1 Ratings-Based Representation

The ratings-based representation consists of the ratings given to the item by the user using a rating scale. The user's ratings are stored so correlation techniques can be used to find similar users.

2.6.2 Content-Based Representations

Content-based representations store representations of specific items of user's inserts so machine-learning techniques can find similar items. Most content-based analysis is performed on textual documents, such as web pages and document abstracts (Middleton, 2003).

The most common abstraction of a textual document in the machine learning context is a **term-frequency vector.** Terms consist of single words, and the frequency count is the number of times a term appears within a document text. To improve processing efficiency, common terms (called stop words) and low frequency terms are removed, since they have little discriminating power. The removal of stop words is normally performed using a standard stop list. Term-frequency representations are often called "bag of words" representations.

The term frequent representation is not able to distinguish between the long and the short documents, and this is a major problem in this approach. Recently a modified version of this

approach has normalised the term frequency by taking into consideration the length of the document.

The most common profile representation for content-based recommender systems is the **binary class profile.** The binary class represents the active user interest as a set of positive and negative examples. The positive examples are represented as a collection of term-frequency vectors of documents that the active user has rated as interesting, while the negative examples are likewise represented (Middleton, 2003).

2.6.3 Knowledge-Based Profile Representation

This approach requires questionnaires and interviews with users to acquire information about their requirements before a profile can be built.

2.7 Profiling Techniques

2.7.1 Time-Decay Functions

Time decay functions are simple profiling techniques which can be applied to profile representations as long as their information is time-stamped. A weighting function is defined which contains an inverse weight, then it is applied to one of the rating values. This weighting function ensures that older information is less relevant than more recent information (Middleton, 2003).

$$w(t_i) = \sum_{j=1}^{N} \frac{tf(t_i,d_j)}{age(d_j)}$$

$w(t_i)$	weight of term t_i after time decay
t_i	i^{th} term
N	number of documents
$tf(t_i,d_j)$	number of times term t_i appears in document d_j
d_j	j^{th} document
$age(d_j)$	age of document d_j

Figure 2.3: Time Decay Functions (Middleton, 2003, p. 33)

2.7.2 Data Mining Techniques

In building a user's profile, data mining techniques will automatically discover and extract information from the data present in web pages which can be used to discover patterns of behaviour in a large dataset of user behaviour. Web mining is divided into three active research areas: content mining, structure mining and usage mining (Dunham, 2003).

Content mining is the process of extracting useful information from the web content, including documents and other underlying link structures. Structure mining uses links and references within web pages to obtain the underlying topology. In contrast, usage mining performs mining on web usage data, for example the behaviour of web users when accessing the website, in order to extract interesting usage patterns. The usage mining will detect the users' behaviour, which will be used to build the users' profile and provide adaptive services. Moreover, the users' profile can be developing by studying their patterns on accessing web pages. Once developed it can be used to provide users with more personalised services.

2.8 Recommender Systems in Practise

In this section I will give a brief description of the most popular existing recommender system features.

- **Amazon.com:** The Amazon website has a large amount of information that consumers must process in making their choices. Recommender systems are used on the Amazon website to personalize the online store for each consumer and to suggest products to consumers that match their interests. The product can be recommended based on an analysis of the historical purchase of the consumer, which is based on an item-to-item algorithm designed by Amazon (Linden et al., 2003).
- **GroupLens:** This is one of the first recommender systems built using the combination of content and collaborative based filtering. The idea behind this system is to help users to find articles they like based on other similar users' profiles (Group Lens Research Projects, 2007).
- **MovieLens.org:** MovieLens is a research site created by GroupLens research lab in the Department of Computer Science and Engineering at the University of Minnesota. MovieLens is a recommender system using collaborative filtering technology to make

recommendations of movies by matching users with similar movie preferences based on others' ratings. When a new user logs in, the system asks the user to rate fifteen movies to start with. Users are asked to rate every movie they watch, and Pearson correlation coefficient algorithm is used to find similar users. When the user logs in to the website the next time, the system will provide the user with a recommended set of movies that the user has not watched, along with their ratings (MovieLens, 2008).

- **Ringo:** This system recommends products based on the opinions of a community of similar users, using collaborative filtering (Shardanand and Maes, 1995).
- **CDNOW:** This is a commercial application for buying music CDs. CDNOW provides a set of recommendations to each user based on the user's buying history and the ratings the user has previously given to music CDs the user heard.

2.9 Summary

This chapter first introduced the problem of information overload, in which the users are trying to deal with more information that does not enable them to make sensible decisions. Then, a recommender system was presented as a solution to this problem which overcomes limitations of searching through the mass volume of information available for the users. The three common algorithm techniques for computing recommendation were discussed along within their advantages, limitations and techniques. Also, profiling techniques commonly employed by recommender systems were discussed. Finally, a brief introduction was given to existing work on e-commerce recommender systems.

3 Requirements

This chapter presents the requirements of the system, which are based on the project scope described in section 1.3. The requirements have been defined from the analysis of research conducted with similar existing recommender systems. Although this thesis is focused on design and evaluation of recommendation algorithms, it is important to describe the basic requirements of the system.

3.1 System Overview

This project involves designing and evaluating a recommender system within the book domain based on the project scope. Different kinds of recommender system algorithms will be designed, implemented and tested with real end-users. The system will build users' profiles by evaluating and tracking their behaviour on the web site. After that, the system will produce non-personalised and personalised book recommendations and provide these to the user.

3.2 Functional Requirements

The functional requirements define the main functionality of the system. They are ordered here according to their importance, the most critical (and those that are to be developed first) are listed first.

3.2.1 General System Requirements

GR-001 Registration
A guest should be encouraged to register in order to take advantage of benefits only open to members, such as receiving personalised recommendations from the system. The registration process will capture personal details, login information and a knowledge base about the user's preferences and interests.

GR-002 Login/Logout
In order to login to the system, a guest must be a member of the web site and hold a valid username and password.

GR-003 User Control Panel (UCP)
Upon a successful login the user will be presented with their own User Control Panel (UCP). A UCP will encapsulate the following functionalities: the user's favourite books, user's shopping cart, statistical data about the user's preferences, user's profile (which is responsible

for capturing the user's interest through its knowledge based form) and a list of recommendations for the user.

GR-004 Browse
All users will be able to browse books in a certain category or a subcategory within a category.

GR-005 Search
All users will be able to search for a book. There will be two types of search: simple and advanced.

GR-006 Search by Author Name
The automatic creation of hyperlinked authors' names will feature on a book's details. This will enable one book to be related to another.

GR-007 View book details
A book information details will be shown whenever a user request for it.

GR-008 Ratings
The member will be able to rate a book in a numeric scale. The ratings could be updated or deleted by the user who generated them.

GR-009 Add to Favourites
The members will be able to add any book to their favourite books list.

GR-010 Add to Shopping Cart
The members should be able to add any book to their shopping cart.

GR-011 Add to Owned Book List
The members will be encouraged to add any book that they own to their owned book list in order to improve the recommendations.

GR-012 Build User's Profile
The system will capture the interests of the user automatically when a user logs on to a web site. The profile should adapt to changes in user interests over time.

GR-013 Shows Personalised Recommendation
When a member asks for recommendations, the system will display four lists of recommended books based on different algorithms.

GR-014 Shows Non-Personalised Recommendation
Any guest can receive non-personalised recommendations from the system.

GR-015 Improve Recommendation Area
The user may want to exclude a book that had been purchased, added to favourites, browsed or rated from being used in making recommendations. The improve recommendations area will encapsulate the following functionality: favourite books, books the user owns, rated books and books appearing recently in the user's browsing history. This will allow the user to delete any book that is no longer of interest.

3.2.2 Recommendation Module Requirements

3.2.2.1 Non- Personalised Recommendations

RR-001
When the detailed information for a book is browsed, a recommended books list will be provided, based on what the customers have purchased in the past with the browsed book, as well as on what they have browsed.

RR-002
When a user searches for or browses for a book, the system should show the average book rating along with the number of users who have ranked the book using the Mean algorithm.

RR-003
Any user will be able to view statistical data in the book details page that is based on all viewers' rankings of books (on a scale from 1 to 5). Any user will also be able to see, for each ranking of a book, the number of users who gave it this ranking.

RR-004
Non-personalised recommendations will be provided on the website homepage, which will be updated every time the page is browsed, based on the month's bestselling books and the most recently viewed books for that month.

3.2.2.2 Personalised Recommendations

RR-005
The system will provide book recommendations based on collaborative filtering. The system will implement the User-User Nearest Algorithm using two different similarity measures

(Pearson Correlation coefficient and Mean Squared Difference) to evaluate the best similarity measure for the application domain.

RR-006
The system should provide a list of book recommendations to the user based on content based filtering. This should be done by using the user profile, which builds as a result of the user's interaction with the system.

RR-007
The system should provide book recommendations based on the combination of content based and collaborative filtering.

3.3 Non-Functional Requirements

The non-functional requirements describe the actual operation of the system and define the qualities of the resulting system.

3.3.1 Interface Requirements

IR1 The interface will be defined using XHTML for the content and CSS for specifying the layout and style. The CSS will be defined in an external style sheet.

IR2 The interfaces will have a uniform design throughout the site, and user help should be provided whenever requested.

IR3 The application developed is expected to be used by non-computer specialists; therefore it must be simple and easy to navigate.

3.3.2 Performance Requirements

PR1 The system shall ensure accuracy and consistency of the required services. This will be achieved by extensive testing before each deliverable is 'signed off' as completed, coupled with the usage of exception handling techniques.

PR2 The system should produce accurate recommendations that match the user's preferences.

PR3 The database should be capable of handling a potentially large number of users.

3.3.3 Information Security and Privacy Requirement

SR1 The application will be tested on a local host. In case the application is going to be

placed on a public host, software tools as well as hardware components will have to be used to ensure protection from malicious users.

SR2 The system will also provide user authentication, authorisation services and access control.

SR3 The system should include a user authorisation where users must identify themselves using a login user name and password.

3.3.4 Operational Requirements

OR1 The system will be developed using SQL Server Express database with a combination of ASP.NET and C# for the programming language.

3.3.5 Development Methodology

MR1 The system should be developed using an incremental approach. The increments should be tested before being integrated with other increments and before the release of the complete system.

3.3.6 Domain Constraints

Data Protection

DR1 Data held on user's behalf should be held in compliance with the Data Protection Act (1998) in the UK (or equivalent law where this Act does not apply.

Project Deadline

DR2 The project (Software and Documentation) must be completed by Monday 28th August 2008.

3.4 Use Case Model

3.4.1 Use Case Diagram

From the functional requirements described in section 3.2, use cases are derived and relationships between them are defined. The following use case diagram (**Figure 3.1**) shows the main activities a user could perform in the system.

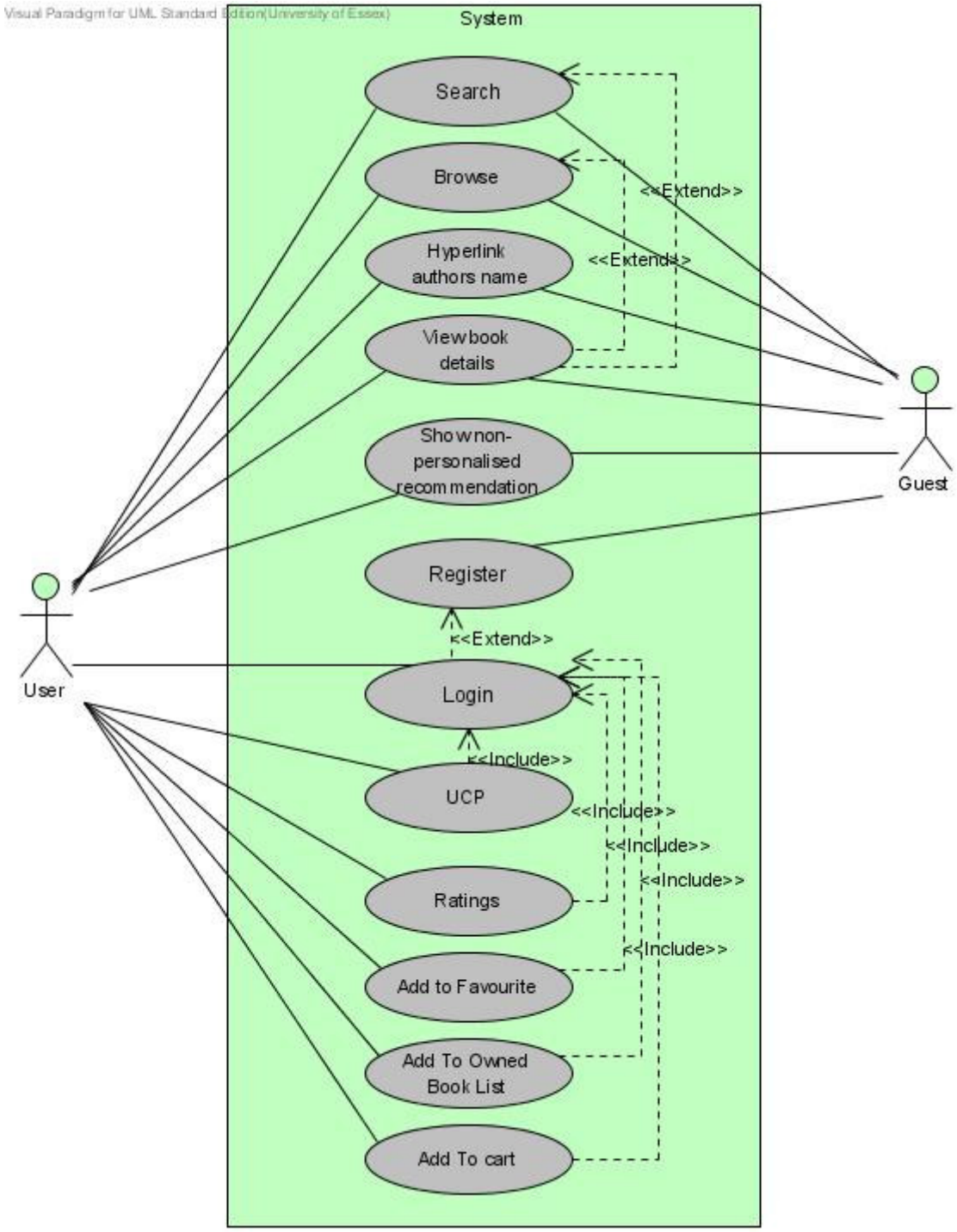


Figure 3.1: Main Use Case Diagram

It assumed that the user can receive different kinds of non-personalised and personalised recommendations based on different algorithms. The following use case diagram (**Figure 3.2**) shows the activities that users can perform in their control panel.

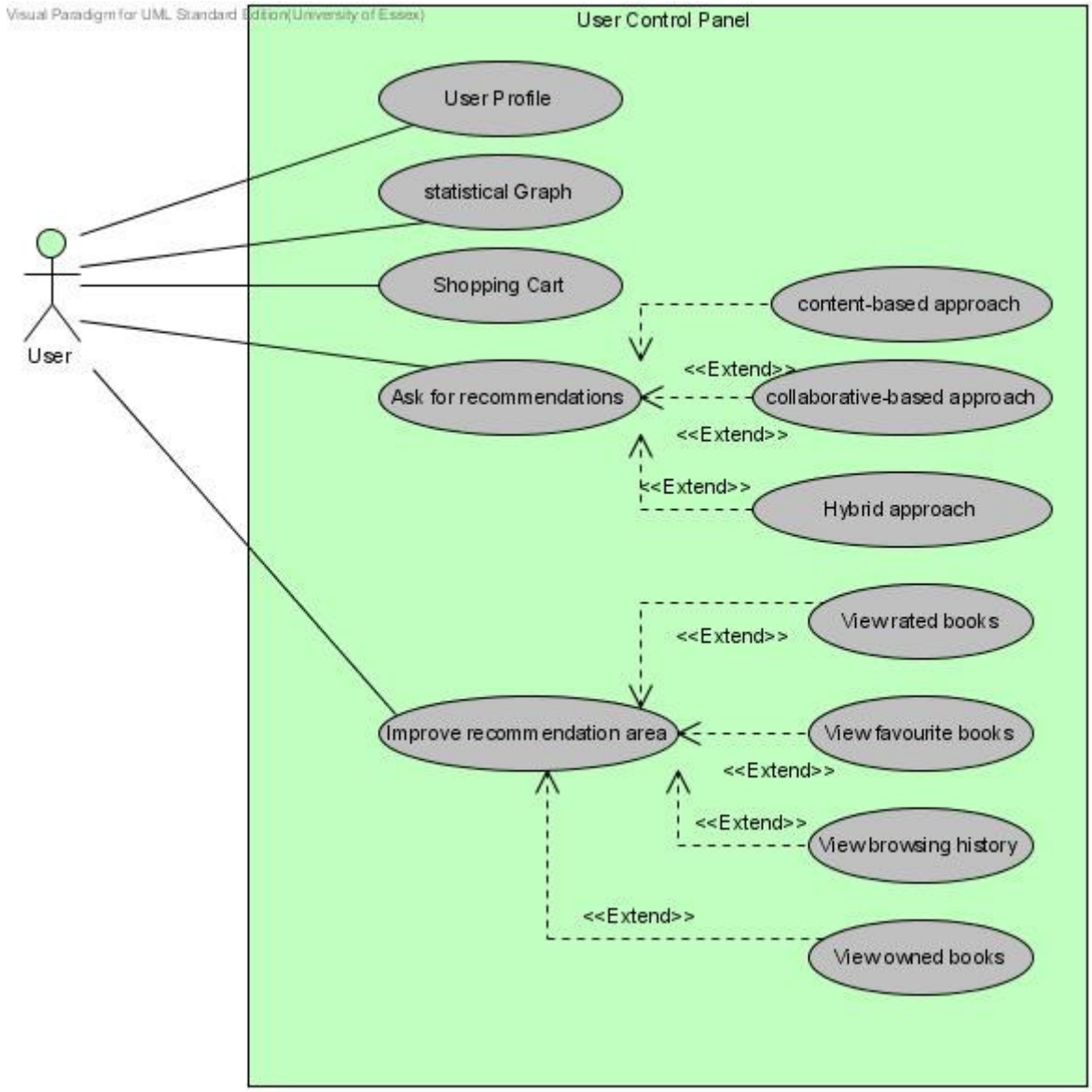


Figure 3.2: User Control Panel Use Case

3.4.2 Use Cases

The use cases have been assigned packages to give a clear structure, as follows:

- Accounts
  - Register
  - Login
  - Edit profile
  - Statistical
  - View shopping cart
- Browsing
  - Browse a category
  - Search
  - View book details
  - Hyperlink author's name
- Learning module
  - Add to cart
  - Add to owned books
  - Add to browsing history
  - Add to favourites
  - Rate
- Recommendation module
  - Personalised recommendations
  - Non-personalised recommendations
- Improve user recommendation area
  - View rated books
  - View favourite books
  - View browsing history
  - View owned books

Due to lack of space, the detailed use cases are provided in **Appendix C**, and the network activity diagrams are provided in **Appendix D.**

4 Design

This chapter presents the design of the system and seeks to satisfy the requirements stated in chapter 3. In addition, this chapter will discuss in detail the design of the recommender system module, along with the proposed algorithms.

4.1 Requirements Mapping

The application will be developed using the incremental development methodology and will be made up of four increments: Front End, Learning module, Recommendation module and Database increment. The requirements outlined in the Requirements Document will be mapped to manageable increments.

4.2 System Architecture

Today, most e-commerce applications adapt three-tier architecture, which has a higher maintainability than the traditional two-tier architecture. The components in the three-tier architecture are well separated, and the interface between components is well defined (Hung and Zou, 2006). The application design was organized into three major, disjunctive tiers as follows:

- Presentation Tier (Front End).
- Logical Tier (Middleware).
- Data Tier (Backend).

The reason behind my design is that each layer can be deployed in physically separated machines, which allows for parallel development in the implementation phase. In addition, reusability of the business logic component is greater for the presentation layer, and maintenance of the business logic is easier. The characteristic of the tier communication is that the tiers will communicate only with their adjacent neighbours.

ASP.NET web forms make up the presentation tier. Each web form in ASP.NET consists of two separate files which separate the code from content. The content page has an .aspx file extension and contains all the HTML and ASP.NET server controls. The code page has an .aspx.cs extension and contains the code for the page. One of the advantages from the code-behind model is that there is no need to declare public variables in the code-behind page to reference controls in the content page; this some times caused problems if controls were removed or renamed.

In contrast, the business tier consists of Business Objects and Data Access Objects. The Business Objects consist of C# classes, each of which represents an object type in the system. The Data Access Objects consist of one or more C# classes which can receive and process requests from the presentation layer. Also, these classes can request data from the Data Store Layer, which consists of SQL Server Express using SQL commands or prepared statements in the database. Moreover, the Data Access Objects consist of three main classes: The DbConnection class manage the connection to the database so other classes can interact with the database. Class DbUpdate stores most methods that are related to the update contents the system would like to change. Class DbQuery, apparently, interacts directly with the database in order to extract data needed for display or processing.

The following diagram demonstrates this architecture; arrows represent the communication between layers.

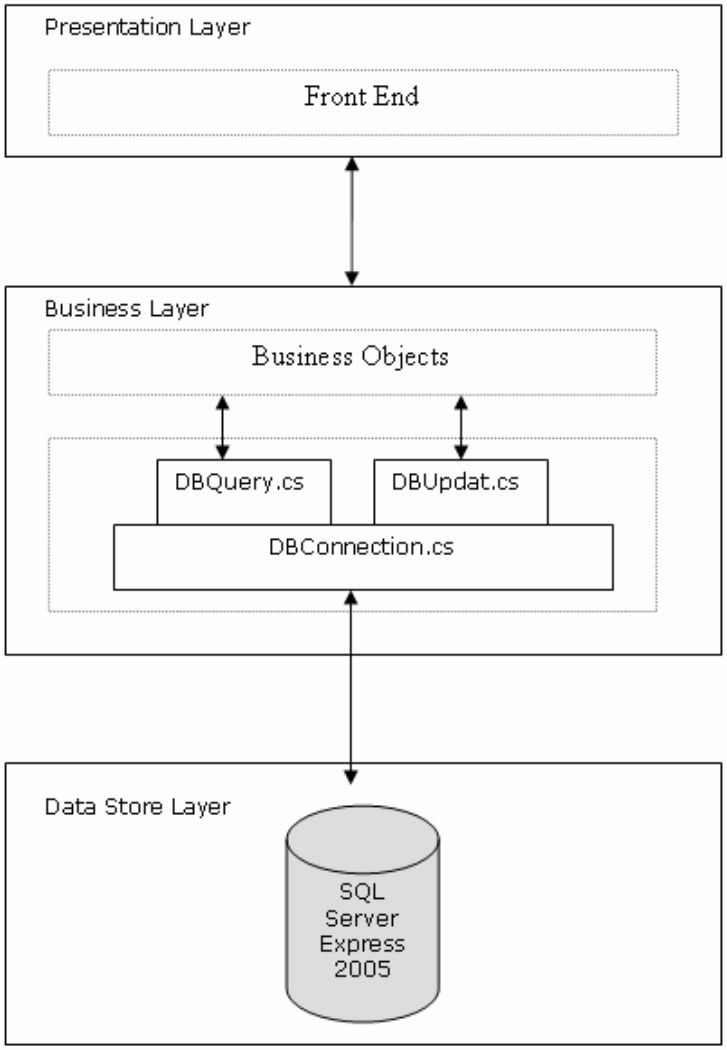


Figure 4.1: Three-Tier System Architecture

4.3 Design Pattern

The first thing to be done in the application design is to build the web application based on a known design pattern, and then extend it based on the system requirements. Design patterns can help solve complex design problems. In my design, I have chosen to use Model-View-Controller (MVC) design pattern, **Figure 4.2**, because it improves reusability of business logic by separating the three components required to generate and manage a specific user interface.

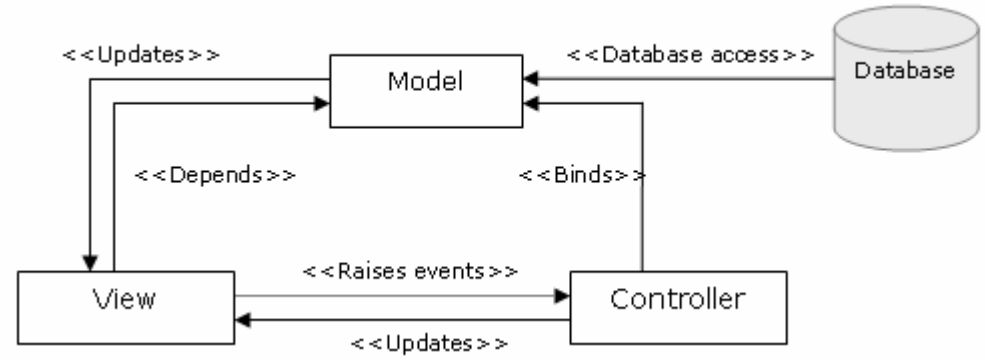


Figure 4.2: Model-View-Control Design Pattern

ASP.NET provides a perfect way for implementing the MVC design pattern. The patterns for the view and controller are well defined, and the model is left to the developer to design (Valenzuela and Fawcett, 2008).

- **View:** The view raises events in the controller, which updates the model. The aspx and ascx files handle the responsibilities of the view.
- **Controller:** The controller functions are implemented in the code-behind file, which contains all the logic and processing code. The controller functions adapt the model code to the data controls that exist on the page. In addition, event handlers within the code-behind file handle events raised by controls to perform actions when a post back to the server occurs.
- **Model:** In ASP.NET the developer has the option to create a model class. In my design, the code-behind file will use a separate data access layer that performs the model functions to read from, write to, and expose the source data.

The main objective of the MVC design pattern is separating the model from the ASP.NET environment. This makes testing of the model code easier and reduces the code duplication.

For example, one class can now be used by many pages. This eliminates the need to copy the code into multiple pages. In addition, the MVC design pattern promotes better code organization, scalability and code re-use.

4.4 Increment Design

This web application consists of four increments: Front End, Learning module, Recommendation module, and Database increment. Each increment represents a subsystem (see **Figure 4.3**).

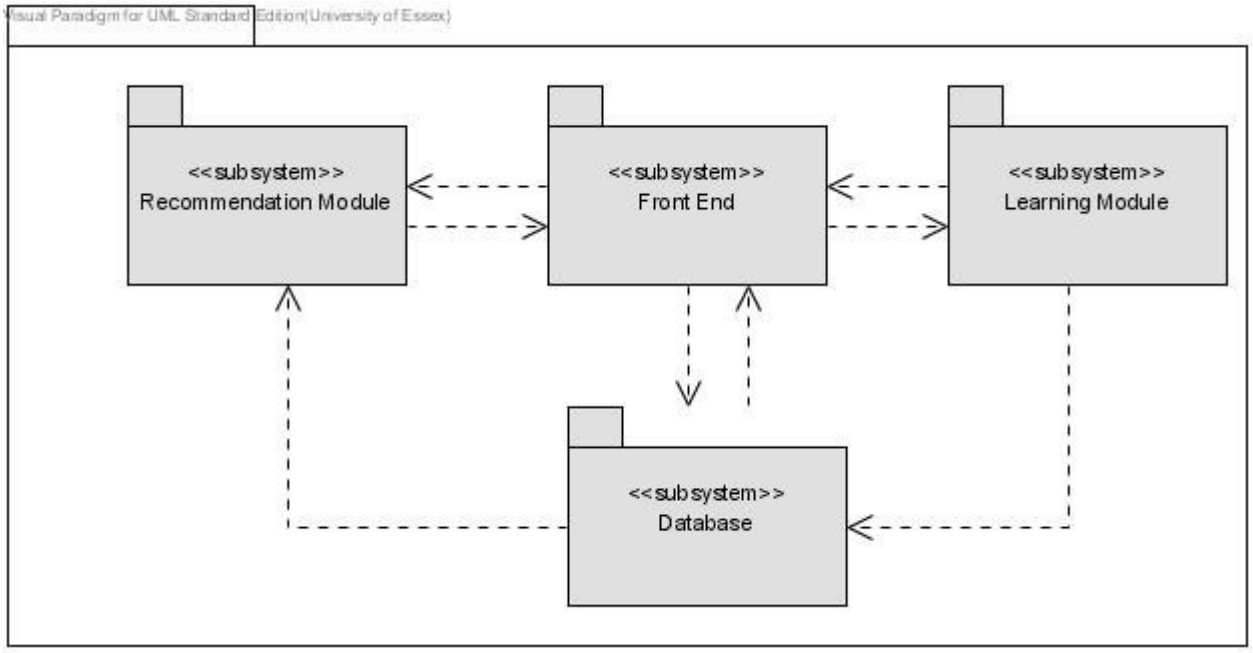


Figure 4.3: High-Level Package Diagram to Model Subsystems

4.4.1 Increment One: Front End

The Front End increment of the system includes the general operations a user could perform in the system. In addition, this increment encompasses the learning module for building users' profiles, as well as the recommender system algorithms, and deals specifically with the presentation of the online book shop. It is responsible for the main operations of the system, which include registration, login procedure, search books, browse a particular category, view book details and user control panel. This increment will be developed using a combination of C#, HTML and CSS.

User Interface

Through the design of the web application page layout, I have tried to keep the presentation user friendly whilst providing the functionality required to make users more likely to return to the site. The website must display as intended in a variety of web browsers and with various versions of the browser. In addition, the choice of colours and the availability of fonts on users' computers are important aspects for the website design.

A header and footer with two-column page layout will be used for the application page layout. A master page will be used, as the page frame which will contain common elements on every page. In addition, the master page will have a main container populated through the use of an ASP Content Placeholder component in which other pages can be placed so that the master page and content page are combined into one page. The layout will be styled using a combination of XHTML, an external CSS and some ASP controls. The structure of this website will be achieved through a combination of some embedded tables and div tags.

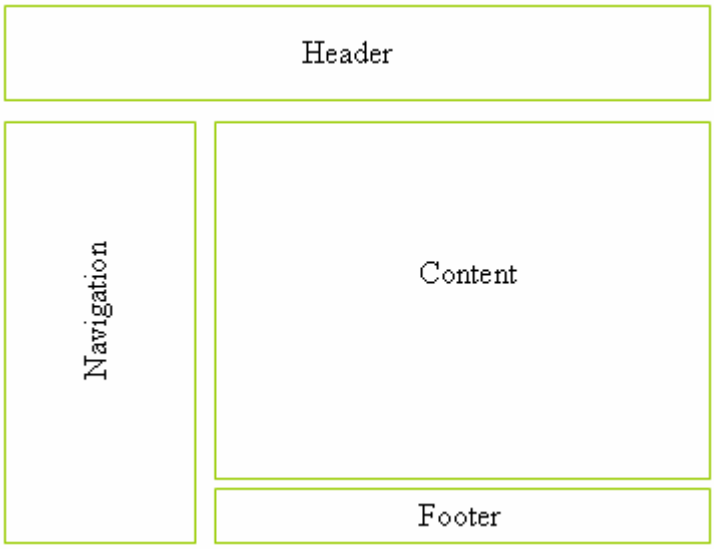


Figure 4.4: The Page Layout of All Pages in the Application

The above figure is split into four sections the web site logo, the navigation, the footer and the main content; this makes it easier to display the functionalities in a clear and consistent way. The navigation will be divided to three sections: the search functionality, links to major areas of the web site (such as logging in and the web site home page), and the navigation itself, which contains controls for browsing categories. Each navigation category will have its own custom navigation. The reason for displaying the navigation in this way is to help the user understand the hierarchical nature of the categories and navigate easily through the different pages.

4.4.2 Increment Two: Learning Module

One of the important issues of the recommender system is how to represent users' interests. This can be done by constructing the model of user in terms of a profile that will be used to represent users' preferences. The user profile attempts to capture the interests of the user automatically when a user logs on to a web site. The profile should adapt to changing user interests over time. However, machine learning techniques will be used to build and maintain users' profiles, while content-based and collaborative filtering techniques will be applied on the user profiles to identify the relevant items/users and then produce the recommendations.

In the application design, the user profile represents a set of categories and, for each category, a set of subcategories, with a frequent value and a bag of words (BOW) which represent the user's interest in that category. The frequency value of the subcategory in a category reflects the significance of the subcategory in representing the user's interest in that category. The BOW forms the words in the book's title and description after removing the stop words that are assumed to have no information content. Each word in the BOW has a weight value that represents the appearance of the word in the text. However, user profile construction techniques can be divided into explicit and implicit feedback, based on the type of input used to build the profile (Gauch et al., 2007).

4.4.2.1 Explicit User Information Collection

Explicit user information collection methodologies rely on the input by users of personal information, such as demographic information and personal interests. Other methodologies that collect users' preferences include techniques that allow users to express their opinions by selecting a value from a range of options. These methodologies have the drawback that they require the user's time and effort. Nevertheless, many sophisticated, personalised commercial systems are based on explicit feedback (Gauch et al., 2007).

In the application design, I will be used six forms to gather users' interest, applying explicit user information collection methodologies:

- **Registration Form:** When the user requests registration, the registration form appears, along with a preference elicitation form that will acquire information about the user interest before a profile can be built.
- **Rating Form:** The user ranks a book in a scale from 1 to 5, according to level of interest. After that, this ranking will be used to modify the user's profile.
- **User's Purchasing History:** This form gathers information about the books that the user purchased in the past.
- **User-Owned Books:** When the users browse a certain book, they can inform the system that they owned the book in order to improve their recommendations.
- **User's Favourite Books:** When the users browse a certain book, they can add it to their favourite list so the system will learn their interests.
- **Preference Elicitation Form:** This form exists in the user control panel. Users may complete it if they want to improve their profile for more accurate recommendations. The preference elicitation form is based on a common feedback technique, that is, to allow users to express their opinions by selecting a value from a range. It gives a simple checklist of all subcategories within each category so the users can express their opinions by selecting a range of subcategories that they prefer in order to improve their recommendations.

4.4.2.2 Implicit User Information Collection

Implicit user information collection techniques do not require any additional effort by the users during the process of constructing profiles. The most popular technique used to collect implicit feedback is to track the users' browsing activity. Capturing browsing activity at the site is a common means of collection information from which users' interest can be extracted (Gauch et al., 2007).

The system will require a history of the user's interactions with the system, which includes storing the browsed book information along with the book category and subcategories that will be used to modify the user's profile using a frequent value.

4.4.3 Increment Three: Recommendation Module

The recommendation module of the system focuses on design and develops different approaches for computing recommendations within the book domain helping users find unfamiliar books that they probably will like. This module is the heart of the proposed system.

The module is composed of the following sections:

- ✓ **Non-personalised recommendations:**
- Mean algorithm
- Non-personalised recommendations based on the browsed book
- Non-personalised recommendations on the website homepage

- ✓ **Personalised recommendations:**
- Content based approach, using the user's profile
- Collaborative filtering approach, using User – User Nearest Neighbours Algorithm
- Hybrid approach, using User – User Nearest Neighbours Algorithm with the combination of the user's profile

- ✓ **Improve users' personalised recommendations**

4.4.3.1 Non-personalised Recommendations

Mean algorithm is the most common non-personalised recommendation algorithm that reflects the common interests of the users within the dataset. The Mean algorithm, sometimes referred to as the item average algorithm or the POP algorithm, as it falls within the popularity predication techniques (Breese et al., 1998). The prediction for an item is calculated as the mean value of the votes of all users for that particular item. The mean score for the item i is defined as:

$$\overline{p_i} = \frac{1}{|P_i|} \sum_{i \in P_i} v_{ij}$$

where p_i is the set of users who have rated item i, and v_{ij} is the user vote for item i.

One of the most important objectives of the recommender system is to improve loyalty by creating a value-added relationship between the site and the customer. As a result of this relationship, customers repay these sites by returning to the ones that match their interests (Konstan et al., 1999). For this reason, three kinds of non-personalised recommendations will be found on the information page for each book, based on an average of other users' opinions about the book. This page in fact consists of two separate recommendation lists and one statistical graph. The first list recommends books frequently purchased by customers who purchased the browsed book, while the second one recommends books frequently viewed by customers who browsed the selected book. In contrast, the statistical graph located on the information page for each book represents statistical data based on 1-5 star ratings; for each star rating, a number of users who provide this rate for the book will be shown.

The website home page will display the most popular books based on the bestsellers book, and the books most frequently browsed by the users, for the month. These two recommendations are independent of the user, so each user gets the same recommendations. Non-personalised recommendations are automatic and ephemeral because they require little customer effort to generate, and the system does not need to recognize the user (Konstan et al., 1999). In addition, the system will continually adjust the bestsellers of books against each other; the bestselling books displayed will be updated each time the page is browsed. These two non-personalised recommendations are intended to be of general interest and may help the customer in making a decision about books that they originally held in doubt. They are not meant to improve cross-sell by suggesting additional books for the customer to purchase.

4.4.3.2 Personalised Recommendations

Content-Based Filtering

Producing recommendations for a user based on content-based filtering is done by selecting a set of items that match the user's profile. To implement this approach, the system must understand the item domain well, and have knowledge of all the item features. In my design, the system will provide the users with books within the subcategories that they preferred in the past. It will work by tracking the user's behaviour, and gather as much evaluation data as possible from the users. The combination of the user's behaviour and the evaluated books will help the system to build the user's profile. However, in keeping with the content-based design approach, I have chosen to provide the active user with books within the same

subcategories as the user previously preferred. The reason behind my design is that one book category may contain a considerable variety of subcategories.

The first step in the content-based filtering approach is to generate a list of subcategories within each category from the active user profile in which the frequent value is higher than a threshold. After that, the recommendation module takes the list generated from the first step along with the books that the active user did not evaluate from the database, in order to compute the degree of interest for each book. Finally, the recommendation module creates a list of books and then provides the active user with recommendations.

Collaborative Filtering

Recommendations in collaborative filtering are based on the correlation between users with similar interests. The User–User Nearest Neighbours Algorithm is the basic algorithm in collaborative filtering approach. The algorithm uses statistical techniques to find a set of users (known as neighbours) with interests similar to those of the active user in the database. Then, the algorithm uses the rating's information of the most similar users to compute a prediction for the books rated by the neighbour users, to generate a list of recommendations for the active users. However, the algorithm will only compute a prediction for the books that the active user has not evaluated. So the algorithm will help users to find unfamiliar books that they probably will like.

The user-based approach avoids an expensive model-building stage, but it has some disadvantages, as it requires a large number of users and items; this causes a scalability problem (Sarwar et al, 2001). In addition, this approach also results in a problem; it can not recommend new items that have not been rated by any user. This is called the "cold start problem" (Middleton, 2003).

In my design I have chosen the user-based approach; because of the limited number of users and books in online book shop, the scalability issue is not a problem. The cold start problem may be an issue; the number of users and evaluations will be small compared with the number of books, which may make it more difficult to produce recommendation. I will try to solve this by first using the content-based approach to gather some evaluation data before introducing the user-based approach.

The process for producing recommendations can be divided into three steps (Fasli, 2006):

1. Compute the degree of similarity between the active user and the other users in the database, based on the common rated books between them.
2. Select, from the neighbours list generated in the first step, the subset of users with the highest similarity.
3. Generate a set of recommendations based on the selected subset of users and calculate the prediction for each recommend book in the recommendation list.

The first step is to compute the degree of similarity between the active user and the others in the database, on the basis of the ratings previously collected. In my design, I will use two different measures of similarity that are based on neighbourhood algorithms: Pearson coefficients and mean squared differences. In this way I can evaluate the two algorithms and show which one is more accurate. The two algorithms formulae and details of how they work can be viewed in section 2.5.1.

In the second step, the system will select a subset of users as neighbours based on the degree of similarity from the previous step to guarantee accurate recommendations. The reason behind this is that many of the users do not have tastes similar to those of the active user; therefore, selecting them as neighbours will decrease the accuracy of the recommendations. Two techniques have been used by researchers to determine how many neighbours to select: *correlation threshold* and *best-n-neighbours* (Mortensen, 2007). The first technique sets an absolute correlation threshold; all neighbours with a degree of similarity greater than this threshold are selected. The problem with this technique if the threshold is set too high, then few users will be selected as neighbours. The second technique selects the best *n* neighbours for a given *n* number, based on the degree of similarity. This technique solves the problem with the first one, but it introduces another problem because the top neighbours do not always provide a perfect basis for producing recommendations (Mortensen, 2007). However, I have chosen to use the correlation threshold for selecting neighbourhoods for each user. The reason behind this is that when I set an appropriate correlation threshold for the application domain, the problem will be solved and accurate recommendations will be produced. In contrast, the problem with the best-n-neighbours technique would not be resolved, due to the small number of neighbours to the active user in my proposed system.

The third step is to compute recommendations for the active user based on the selected neighbourhood and the ratings from users in the neighbourhood. This will be done by traverse the neighbourhood list, starting with the most similar user. All books that the neighbourhood user has rated positively and that the active user has not yet rated will be added to the recommended books list. Finally, the list will be filtered by computing the prediction for each book in the recommended books list. The prediction on item *j* for the active user *a* is computed by picking *k* neighbour users who have also rated item *a* (Rashid et al., 2005).

$$p_{aj} = \overline{v_a} + \frac{\sum_{i=1}^{n} w(a,i)(v_{ij} - \overline{v_i})}{\sum_{i=1}^{n} |w(a,i)|}$$

where v_a is the mean of the active user votes, n is the number of users in the data set with non-zero weights, $w(a,i)$ is the correlation between each user, v_{ij} is the rating for item j and v_i is the mean of all user votes with non-empty votes (Breese et al., 1998). If the book prediction is over the threshold, then the book will be recommended to the active user.

Hybrid Approach

The User-User Nearest Neighbours Algorithm with user's profile combines content-based and collaborative filtering recommendation techniques. The algorithm works by creating a list of the most popular book categories in the user's profile for which the frequent value is higher than a threshold value. After that, the algorithm creates another list containing the top five subcategories in each category based on the categories in the previous generated list and the subcategories in the user's profile. For example, if the user has evaluated more database, web design and programming books in the computers category, the user profile is built including these subcategories. Then, when a prediction for a recommend book is calculated and it matches with one of the categories generated from the user's profile, the algorithm adds one point scale to the prediction. Also, if it matches with one of the subcategories generated from the user's profile, the algorithm adds another point scale to the prediction.

The following steps will be added to the common User–User Nearest Neighbour algorithm:

4. Generate a list of the most popular categories and subcategories from the user's profile.

5. Include an extra value into the prediction calculation, which combines the user's profile with the collaborative filtering algorithm.

4.4.3.3 Improve user's personalised recommendations

Improve user's recommendations will help the system to improve the user's profile by observing and evaluating changes in the user's interests and preferences. Moreover, the system will help users to keep track of books they have recently evaluated in the *Improve Your Recommendation* area to produce more accurate recommendations.

4.4.3.4 Domain Class Diagram

The class diagram (**Appendix E**) shows the classes and their relationships in a subsystem. It shows the most important classes on the Recommendation module subsystem. This was decided due to the complexity of the entities and the relationships within a subsystem.

4.4.4 Increment Four : Database

The Database increment is the most important increment within the system because it stores the data needed to allow the recommendation algorithms to function as intended. The design of the database focuses on the learning module and how it will be used by the recommender system. Moreover, no normalizations will be provided for the table that contains information about books details such as provides separate tables for the book's authors and publisher. This was decided for two reasons: first, the purpose of this thesis is to design and evaluate a recommender system within the book domain, not to build an efficient online book store. Second, the recommender system does not need information about the books' authors and publishers to produce recommendations for the user. In addition, the project will use the book ID as a primary key for the book dataset, rather than the book's ISBN. This is because the data set will build, "from scratch" different sources; as a result, I am not sure that I could provide accurate data that did not duplicate the data set.

4.4.4.1 Entity-Relationship Diagram

The Entity-Relationship (ER) Diagram (**Appendix F**) is a major data modelling tool and uses specialized graphics. It defines and demonstrates the relationship between entities. In

addition, it will aid the developer in organizing the database structure, in order to enable the developer to store and retrieve data in an efficient way.

The ER diagram only shows the names of the tables, their primary keys and their unnamed relationships. This is because of the complexity of the database. Each relationship represented in the ER diagram is described in more detail in **Appendix G.**

4.4.4.2 Tables and their Attributes

The sixteen tables displayed in the ER diagram are briefly outlined below. Each table's attributes and data types are displayed in full in **Appendix H.**

- The **Users Table** will be used to store user personal details, such as the user's title, name, country, address, birthday, phone, email address and username.
- The **BookDetails Table** will store all the book details to be used in my experiment. The following information will be stored for each book: title, author(s) name, publisher details, ISBN, a brief description, an image, the book's price, a reference to the book's category and the quantity of copies available. In addition, each book belongs to one category and at most three subcategories within the category.
- The **Categories** table will store the category name and ID.
- The **SubCategory** table will store the subcategory name and ID. It will also contain a foreign key to the main category ID to which the subcategory belongs.
- The **BrowsingHistory** table will store the date visited by a user for a certain book, along with the book ID and the user ID, which will used to connect to the user's browsing history.
- The **Ratings** table will store the rating for a given book, along with the book ID, category ID and the user ID for the person who rated the book.
- The **UserFavourite** table will store the favourite book ID and category ID for a given user along with his/her ID, which will used to connect to the user's favourite books.
- The **OwnedBook** table will store the user's owned books ID, along with the book category ID and the user ID which will be used to connect to the user's owned books.
- The **Purchased** table will store the user's purchased books ID, along with the book category ID and the user ID, which will used to connect to the user's owned books.

- The **UserTrash** table will store any book information that the user marked as not of interest, along with the user ID.
- The **Profiles** table will store the user's preferences in a given category.
- The **Countries** table will store the country's name and the country phone code.
- The **StopKeywords** table will store the stop words (and, or, the, etc.) to be used on clustering the book's attributes. In addition, this table will contain 438 stop words from different stop lists obtained on the Internet.
- The **Keywords** table will store the book's title, author name(s) and description words.
- The **BOW** table will store the interesting words for the user and will be associated with the user's profile to identify the category and subcategory to which the BOW record belongs.
- The **Evaluation** table will store the users' evaluations for the different kinds of the recommender system algorithms to be used for the experiment.

4.4.4.3 Domain Class Diagram

The database increment consists of three main classes (**Appendix I**): The `DbConnection` class manages the connection to the database so other classes and subsystems can interact with the database. Class `DbUpdate` stores most methods that are related to update contents in the database that the system would like to change. Class `DbQuery` interacts directly with the database in order to extract data needed for display or processing.

5 Implementation

In this chapter the implementation environment of the recommender system is described. The programming language and the implementation details of each increment of the system are described.

5.1 Implementation Tools

5.1.1 programming Language

Several frameworks and languages can be used to develop the proposed system. A research and investigation was conducted to determine which programming language should be used.

- **Java** is an object oriented programming language developed by Sun Microsystems. It is a common platform for web application development. One of the most significant advantages of Java is that it offers platform independence (The Source for Java Developers, 2008).

- **PHP** is an open-source scripting language and is very popular for web development (TIOBE, 2006).

- **ASP.NET** technology is a unified web platform which is implemented based on .NET Framework to create dynamic web applications. The .NET Framework supports a variety of programming languages, such as VB.NET, C# and Jscript.Net. ASP.NET has powerful tools to build the web applications and web services in a short period and in the easiest way. In addition, ASP.NET provides direct support for web services and provides a configuration system that defines configuration settings for the web server and individual applications. Also, ASP.NET Security architecture works with IIS and the underlying security services provided by the operating system to provide authorization mechanisms (msdn, 2008).

The choice of programming language from among the many options depends mostly on the programmer's programming background and the time available to build the project. Also, I decided to eliminate PHP from the comparison as it is not a fully object-oriented

programming and it is weakly typed; this can lead to bugs that require more effort to detect than the bugs in other languages.

Both Java and .NET languages are good choices for the language development of the project, but .NET languages have the advantage of fast development. In addition, .NET languages gives complete control over garbage collection and provide direct support for properties, events, and delegates as part of language. To conclude, it was decided to use ASP.NET technology for this project because it provides far more expressiveness and is a proven way to reduce maintenance and testing costs, whereas Java is not. Also, I chose C# from the .NET languages because it is similar to Java, which I am familiar with.

5.1.2 ADO.NET

ADO.NET interacts with different types of data sources and databases such as Microsoft SQL Server. In addition, it is included with the Microsoft .NET Framework providing access to data stored in relational database systems, XML, and application data (msdn, 2008).

5.1.3 Microsoft SQL Server 2005

Microsoft SQL Server *is a database platform for large-scale online transaction processing (OLTP), data warehousing, and e-commerce applications* (msdn, 2008). SQL Server providing concurrency and a locking system for the database which allows multiple clients to use the same database concurrently.

5.1.4 Cascading Style Sheets

CSS is a style sheet language which describes the presentation of a mark-up document such as, HTML documents. CSS separates the presentation of the application and provides flexibility to future maintenance.

5.1.5 Dundas Chart for .NET

Dundas chart is the industry leader in .NET charting solutions which offers for the developers to add advanced charts to ASP.NET and Windows Forms application (Dundas Software, 2008).

5.2 Incremental Implementation

5.2.1 Database Increment

The Database increment was the first to be developed. The reason for this is to help structure the development of the other increments. The database was created by mapping the tables from section 4.4.4.2 into a relational database.

Whenever a class needs to access data in the database, it must first connect with the database. This was achieved by setting up a connection class which its job to ensure that the database could be accessed from within the other classes. The connection class was implemented to allow their execution to occur anytime the `Openconnection()` and `CloseConnection()` methods are called.

5.2.2 Front End Development

This increment structures the main functionality to be used in both the learning module and the recommender module increments. It has been developed after the database increment and before the learning module and the recommender module increments to ensure that these two increments can be tested during the development. Due to lack of space, the details description of the important functions implementation in the Front End increment is provided in **Appendix J**.

5.2.3 Learning Module Development

This increment responsible for build users profiles by tracks the dynamic changes in their interest. It is important to monition that the application does not provide an inventory system. This was decided because the recommender system does not require inventory to work. The purpose for designing a shopping cart was for the learning module so the user's profile can be improves once the user adds a book to his shopping cart.

The profile of the new user is created during the registration process using a preference elicitation form which will capture the interest of the user. After the user enters valid personal details in the first step of the registration process wizard, a preference elicitation form will be presented in the second step to the user as a check box list which contains books categories and another check box list contains some hobbies the user may be interesting in. When the

user selected categories and/or hobbies that he is interesting in and then fills out valid account details in the last step of the registration wizard, the user's account will be crated successfully and the `AddToUserProfile` method will be invoked for each selected category and hobby to build the user's profile. Moreover, the system will translate the selected hobby to the categories that the hobby represents because each hobby represents one or more category. In addition, theses categories will be added to the user's profile with a frequent value set to one as a started point which represents the significance of the category in representing the user's interest.

The Book Details page will be one of the most accessed pages on the website. This page display the browsed book details and provides non-personalised recommendations to the users. When the user browses a book, the system will automatically add the book to his browsing history by invoking `AddToBrowsingHistory` method. In addition, this page handles evaluation given by the users to capture the users' interest in order to improve their profiles. The evaluation given by the user represented in the following forms: rate the book, add the browsed book to favourite, owned books list or shopping cart. All these functionality work in the same way so I will explain how I implemented the evaluation given by the user in the rating form as an example.

Once the user rate the browsed book and then clicked the submit button, the `SubmitButtonRating_Click` event handler is invoked which checks for the user's rate. If the user previously rates the book, the `UpdateBookRate` method in the `DbUpdate` class is called to updates the old user's rate by the new one. Otherwise, the `SetBookRate` method is called to insert the rate into the database. The record in the `Ratings` table contains the user's ID, the rated book's ID, the rate and finally the date the rate was provided. After that, the user's profile is updated which is the main function of the learning module. This is done by calling the `AddToProfile` method in the `ProfileEngine` class for each subcategory the browsed book belong to in order to improves the user's profile. This method takes the following parameters: the `userID`, `bookID`, `categoryID` and `subcategoryID`. The process for adding to user's profile in the `AddToProfile` method can be divided into three main steps:

1. `AddToUserProfile` method in the `DBUpdate` class is called which updates the frequent value associate with the categoryID and subcategoryID in the active user's profile. This

method calls the `Proc_AddToUserProfile` stored procedure passing for it the `userID`, `categoryID` and `subcategoryID` parameters. The stored procedure first checks the user's profile, if the subcategory within the category does not exists in the user profile, the `userID` is inserted into the `Profiles` table in the database along with the `subcategoryID` and the `categoryID` with a frequent value set to 1 and then a `BOWID` is created for the added record. The `BOWID` is associate with the `BOW` table to represents the interested words for the active user in the sub category that are associate with the `BOWID` identifier in the `Profiles` table. Otherwise, the stored procedure update the frequent value by incremented it by 1.

2. `GetBookKeywords` method in the `DBQuery` class is called which returns all the words along with a frequent value for each word which represents the number of times the word appears in in the book's title or discrption from the `Keywords` table as described in section 1.
3. `CheckBOWID` method in the `DBQuery` class is called to check if the `BOWID` exists in the BOW table.

   If the `BOWID` exists, the `UpdateBOW` method is called which takes the following two paraemetrs the `BOWID` and `BOW` from type `Dictionary<String,int>` where the set of words represents the keys and the frequent value for the word forms the value of the key. This method iterates throw the set of words in the `BOW` variable and if the word exists on the user's BOW matrix, the frequent value for the word in the `BOW` variable is added to the weight value that is associate with the word in the user's BOW matrix. If the word is not exists, the word and the word's frequent value is inserted into the database along with the `BOWID`.

   If the `BOWID` is not exists, the `AddToBOW` method is called which takes as paraemetrs the `BOWID` and `BOW` that genertaed from the second step. This method add each word and it's frequent value form the `BOW` into the database throw the use of `Proc_InsertBOW` stored procedue. The record in the `BOW` table contains the `BOW`'s ID, word and finally the weight value that reflects the significance of the word in representing the user's interest.

5.2.4 Recommendation Module

The primary purpose of this system is to generate different kinds of recommendations. The steps taken to implement this functionality are outlined in this section.

5.2.4.1 Non-Personalised Recommendations

The non-personalised recommendations are dynamic recommendations which are independent of the user, so each user gets the same recommendations.

When a book details page browsed, a search results load or a particular category browsed a representation of the average ratings along with the number of users who rate the book shows with the book details. This is done in the `Page_Load` by invoking the `GetBookRate(BookID)` method which calculates the Mean algorithm to show the average rating for a particular book. Also, the `GetNumberOfRate(BookID)` method is called to retrieve the number of users who rate the book.

Three kinds of dynamic non-personalised recommendations are represented in the book details page.

- **Customer who bought this book also bought**

The `GetBookRecommendation` method produces the first kind of non-personalised recommendations. To find what other books were purchased together with a specific book, I first join two instances of the `Purchased` table on their `UserID` fields while filtering the `bookID` value in the first instance for the ID of the browsed book. Also, I add two rules to the `WHERE` clause: The first rule ensures that only purchased books belonging to the same browsed book category are selected for more accurate recommendations. The second rule eliminates the browsed book from the results list. Then, the query results are grouped by the `bookID` in order to eliminate the duplicate for books that were purchased more than once in the `Purchased` table that contains the browsed book identifier. After that, to get the books that are most frequently bought along with the browsed book, the results are sorted in descending order by how many times each book appears in the result. Finally, the results are filtered by using the TOP keyword to retrieve the top five books of the results; these will be shown in the book details page. The rest of the results can be viewed by clicking the `Found more` hyperlink as shown in **Figure 5.1**.

Figure 5.1: Customer who Bought this Book Also Bought

- **Customers Who Browsed This Book Also Browsed**

The `GetSimilarBrowsingBookRecommendation` method produces the second kind of non-personalised recommendations. This function works similar to `GetBookRecommendation`; however, it conducts the work in the `BrowsingHistory` table rather than the `purchased` Table. Also, this function adds another rule while joining the two instances of the `BrowsingHistory` table on their `UserID` fields. The rule indicates that the two books must be browsed on the same date.

- **Users' Ratings Graph**

The statistical graph located in the book details page is the presentation of ratings data in aggregate for each scale rate. This is done by invoking the `GetCountRate(bookID, scaleNumber)` method to retrieve the number of users who provide a particular scale rate for the browsed book. After these details have updated the variables associated with the retrieved data, the maximum width of a bar in the graph is set by divided the graph width by the number of users who provided a rating for the browsed book. After that, a bar for each rate scale is created. **Figure 5.2** shows an example for the graph in action.

Figure 5.2: Statistical Graph in Action

5.2.4.2 Personalised Recommendations

To make it easier to evaluate the system, the proposed system is implemented in such a way that users can choose the recommendation approach. **Figure 5.3** shows the interface components for the personalised recommendation module, which represents a link between

the active user and other components in the application. This interface allows the user to request and evaluate recommended books.

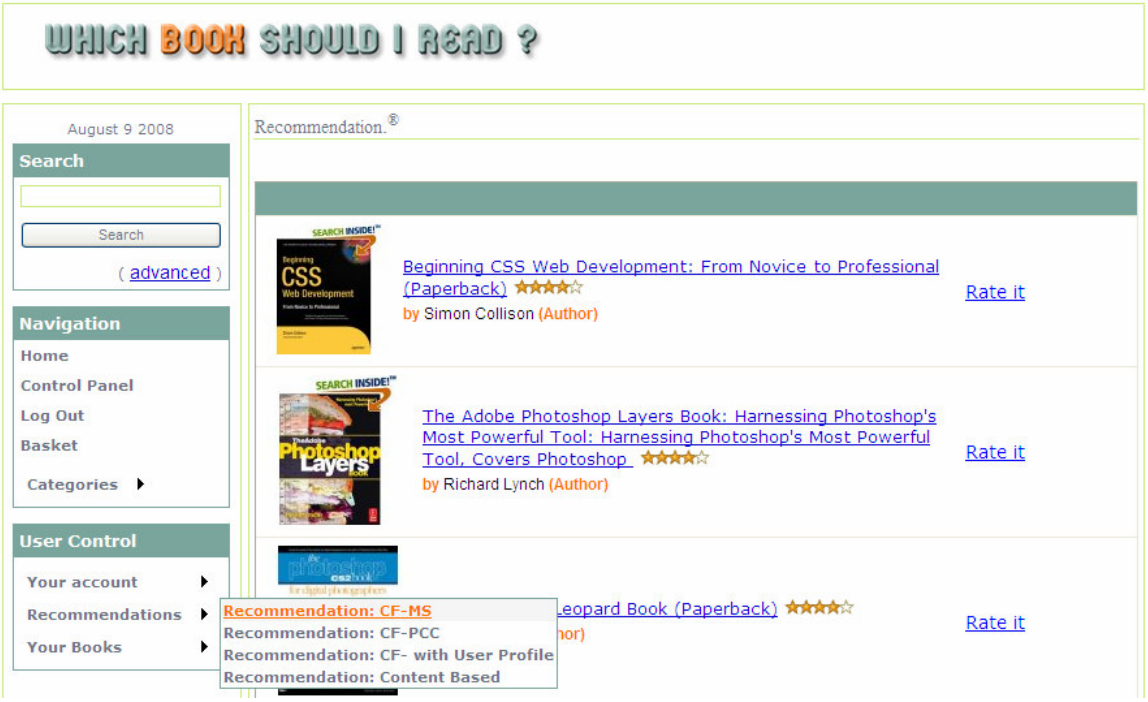


Figure 5.3: Personalised Recommendation Module Interface

Content-Based Filtering

In a content-based approach, when a user asks for recommendations, the recommender first computes the distribution of subcategories in the set of categories on the user profile where the category frequent value is a higher than a threshold value. This results in a list of categories and, for each category, a list of subcategories indicating the user preferences. After that, the recommendation module uses the list generated from the first step, along with the books that the active user did not evaluate from the database, in order to compute the user's degree of interest for each book. Finally, the recommendation module creates a list of recommended books and then provides the active user with recommendations.

Collaborative Filtering

In a collaborative filtering approach, when the user asks for recommendations, the recommender first produces a list containing a specific number of users who have similar interests to the active user. The similarity between the active user and other users is calculated using two different measures of similarity that are based on neighbourhood algorithms: the Pearson coefficients and Mean Squared Differences (both described below). The reason behind this is to evaluate which of the two algorithms will more accurately produce a list of

neighbour users similar to the active user. Both of the two functions take the following three parameters:

- UserID - the active user ID who seeks the recommendations.
- minCommonBook – the minimum number of common rated books between two users to considered neighbours.
- Threshold – the correlation threshold which will be used to filtered the neighbour users list based on the similarity value between the active user and the other users.

• Mean Squared Differences

The degree of similarity between the active user and other users in the database is computed using the Mean Squared Difference algorithm, and implemented using the following SQL query:

```
SELECT r2.userId as neighbor_id, SUM( SQUARE(r1.rating - r2.rating)) /
COUNT (r1.userId) as distance FROM Ratings r1, Ratings r2
WHERE r1.userId=@userID AND r1.bookId=r2.bookId AND @userID !=r2.userId
GROUP BY  r2.userId,r1.userId
HAVING COUNT(r1.userId) >=@minCommonBook
ORDER BY distance desc
```

Figure 5.4: Mean Squared Difference algorithm - SQL Query

The condition ensures that only users that have the minimum number of rated books values or more in common with the actual user are considered neighbours. However, this query returns a similarity list containing each user's ID along with a similarity degree value to the active user. The list is sorted by the degree of similarity, starting with the lowest value, indicating the highest degree of similarity. Then, the list of neighbours is filtered based on the distance value ($msd_{a,u}$) using the threshold value. If both users have similar ratings for all items, the distance value will be less than the threshold value, and the neighbour user will be selected. If the ratings differ, the distance value will be higher than the threshold value.

• Pearson Correlation Coefficients Algorithm

The degree of similarity between the active user and other users is computed using the Pearson Correlation algorithm, which takes into account average ratings for each user. The algorithm implemented by using the following SQL query:

```
SELECT r2.userId as neighbor_id,
(SUM((r1.rating - r1AVG.average) * (r2.rating - r2AVG.average)))/
((SQRT(SUM(SQUARE(r1.rating - r1AVG.average)))) *
(SQRT(SUM(SQUARE(r2.rating - r2AVG.average))))) as distance
FROM Ratings r1, Ratings r2, AverageRatings r1AVG, AverageRatings r2AVG
WHERE r1.userId=@userID AND r1.bookId=r2.bookId AND @userID !=r2.userId
AND r1AVG.userId=@userID AND r2AVG.userId=r2.userId
GROUP BY  r2.userId,r1.userId
HAVING COUNT(r1.userId) >=@minCommonBook
ORDER BY distance desc
```

Figure 5.5: Pearson Correlation Algorithm - SQL Query

This query returns a similarity list containing each user's ID, along with a similarity degree value, to the active user. The list is sorted based on the degree of similarity. Then, the list of neighbours is filtered based on the similarity degree value using the threshold value, in order to increase the recommender accuracy. If both users have similar ratings for all books, the distance value will be higher than the threshold value, and the user will be selected as a neighbour for the active user. If the ratings differ, the distance value will be less than the threshold value (negative value).

The `GetRecommendationUser_CF_UserToUser` method conducts the third step in the collaborative filtering approach. Once neighbours have been selected, a for loop traverses the neighbour list starting with the most similar user. All books that the neighbourhood user had rated positively and the active user had not yet rated will be added to the recommended book list. Then, the list is filtered by computing the prediction for each book in the recommended books list. Moreover, if the book prediction is over zero, then the book will be added to recommendation list. Finally, the list is sorted in descending order based on the prediction value for each book in the recommendation list. This is done by defining static `Comparer<RecommendedBook>` properties in the `RecommendedBook` class, and then supplying the properties as a parameter to the `Sort` method of the list.

Hybrid Filtering

The hybrid approach combines the User-User Nearest Neighbours Algorithm with user profile to produce a list of recommendations. After a prediction for a recommend book is calculated and it matches with one of the top categories generated from the user's profile, the algorithm adds one point scale to the prediction. Also, if it matches with one of the top subcategories generated from the user's profile, the algorithm will add another point scale to the prediction.

5.2.4.3 Improve Users' Personalised Recommendations

The main function of the system is to produce accurate recommendations by tracking the dynamic changes in the interests of the user. The user's interests may change over time; the users can change the record of their old interests by visiting the *Improve Your Recommendation* area. This area encapsulates the following functionality: it shows books rated by the user, the books that the user owns, the user's favourite books and books recently browsed by the user. All these four functionalities work in the same way, so I will explain how I implemented *show rated books* for the active user as an example. **Figure 5.6** shows an example of the *improve your recommendation* area in action.

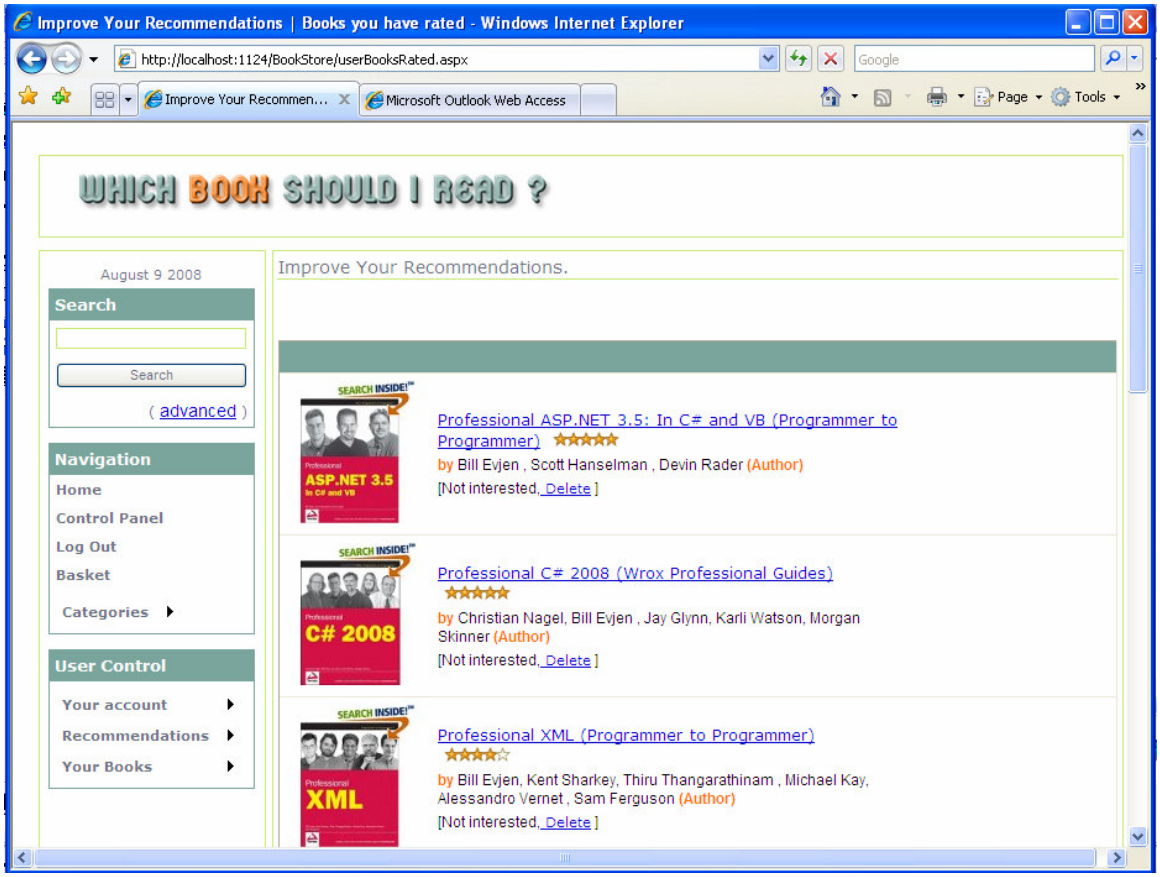


Figure 5.6: Improve Your Recommendation Area in Action

When the user clicks the *Rated Books* link, the `Page_Load()` method is invoked which accesses `GetUserBooksRate(userID)` in the `DbQuery` class to retrieve all the books rated by the active user. If there are no books rated by the user, the system informs the user by displaying a message. Otherwise, the system displays a list of the books rated by the user. If the user is not interested in a particular book, he will click the *Delete* link for that book, which invokes the `GridView1_RowDeleting()` method to delete the book from the user's rated

books list by accessing the `RemoveBookRated` in the `DbUpdate` class. After that, the `RemoveFromUserProfile` method in the `DbUpdate` class is accessed to update the changes in the user's profile based on the deleted book information. Finally the deleted book is added to the `UserTrash` table by accessing the `AddToUserTrash` method in `DbUpdate` class to avoid recommending this book to the user.

6 Testing & Evaluation

This chapter aims to evaluate the quality of the system and determine that it meets its required results. Testing is an essential phase in the recommender system, especially where the system has been released for commercial use. Businesses are paying to use the recommender system to help customers find desired items, and discover new items that they would like to purchase in order to increase their sales. In this instance, erroneous behaviour will have the effect of making users losing confidence in the web site services and take their business elsewhere. It is necessary to conduct testing before the system is used by the end users because it will not just affect the software but also how the business is perceived.

6.1 Application Testing

During the development of the application, three different types of testing were conducted: unit testing, integration testing and system testing.

6.1.1 Unit Testing

During the implementation, individual components in each increment were tested by the developer to reduce the complexity of the overall testing activity and make it easier to detect the components that caused any fault. The test cases were based on the black-box testing method, which considers each component as a black entity and classifies possible inputs into the component. The methodology is only concerned with the input and the result of the tested components. The unit testing results were documented for the classes with the highest probability of usage in the system (**Appendix K**). Please note that all the classes were tested, but the results were not documented due to time constraints and the size of the web application.

6.1.2 Integration Testing

This phase involved integrating the four increments to increase the reliability of the system. The integration testing detects faults within integration components that can not be found from unit testing. The technique used was bottom-up testing, which tests components of the lower level first and then integrates them with higher level components. This was decided because this technique allows interface faults to be easily detected. I started with integrating two system increments and testing them together. If there is no fault, additional increments

are integrated to test. The testing process was repeated several times to ensure that the system worked well after the integration of the four increments. Following that, the testing procedure proved all the system increments were working together as they should. The integration testing documentation for the system's important functions can be found in **Appendix L**.

6.1.3 System Testing

Once the integration and unit testing activities have been completed, the system testing starts. Three kinds of system testing were performed: functional testing, performance testing and finally pilot testing

Function testing is used to demonstrate that the system delivers the functionality as stated in the requirements which is required to operate the web application. When conducting this test it proved that all the functional requirement outlined in **section 3.2** have been satisfied.

Performance testing is essential for any web site to be successful. It consists of the evaluation of the response time of the operations of an application. My focus was mainly on the speed at which recommendations were produced, as the performance of a recommender system is an essential feature. I performed my test in two groups of users; the first group had a rate of more than 30 books, while the second group only rated 10 books. **Figure 6.1** shows the results.

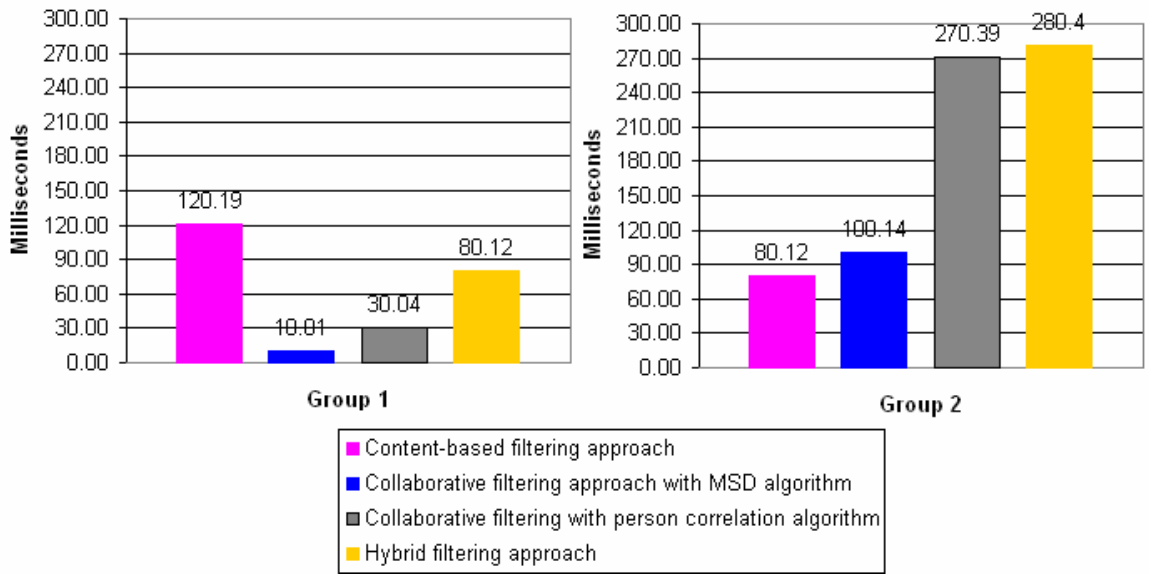


Figure 6.1: Performance Test of Recommender System Techniques

As we can observe from **figure 6.1**, the number of books evaluated by the user had a significant impact on the system's performance. The more users evaluate books, the faster the system will produce recommendations for them. In both cases the collaborative filtering approach with MSD algorithm had a faster performance than collaborative approach using the Pearson correlation algorithm. Also, collaborative filtering is faster for producing recommendations than the hybrid approach. In contrast, content-based approach had a better performance for the group of users who had evaluated 10 books or less.

The pilot testing is the final part of the testing activity. A group of users were selected to test the system, and then the developer received feedback from those users. In the pilot testing, after having tested the system, the users are required to rate the books recommended for them, in order to evaluate the accurate of the system. The majority of the users liked the system's recommendations. Many users suggested providing the user with explanations for why a particular book had been recommended.

The major bug identified during pilot testing is that no recommendations were produced using the collaborative filtering approach. This was due to the small number of users in the system; which has the impact of the difficulty of having similar users in the system for the active user.

6.2 Evaluations of the Algorithms

To evaluate a recommender system, it is important to understand the goals and tasks for which it is being used. One of the main goals of a recommender system is to produce accurate recommendations for the active user.

6.2.1 Evaluation Metrics

Evaluation metrics measure how close the predicted ratings produced by the recommender system are to the true user ratings. For example, MovieLens, a movie recommender system, computes the predicted rate that a user will give each movie and then displays it to the user. After that, the predictive accuracy metrics will compare the MovieLens's predicted ratings with the true ratings users gave to each movie, in order to evaluate the accuracy of the predicted ratings.

Recommender systems research has used several types of measures for evaluating the accuracy of a recommender system. They can be categorised into two classes (Sarwar et al., 2001):

The first of these classes is statistical accuracy metrics. These evaluate the accuracy of a recommender system by comparing the actual rating a user gave to an item with the prediction made by an algorithm. Mean Absolute Error (MAE) between ratings and predictions is a widely used metric. The lower the Mean Absolute Error, the better a predictor can be classified. The following formula forms the Mean Absolute Error (MAE), where pi is the predicted rating and ri is the true one:

$$MAE = \frac{\sum_{i=1}^{N} | p_i - r_i |}{N}$$

The second one is Decision support accuracy metrics evaluate the effectiveness of a prediction engine at helping a user select the appropriate item from the set of all items. These metrics assume the prediction process as a binary operation. The items are predicted, using decision support accuracy metrics, as (good) or (bad).

I used MAE algorithm as an evaluation metric for the evaluation of the predictions made by the system algorithms because I used a numeric scale for the prediction process, not a binary operation.

6.2.2 User Experiments

The system was available to a community of users for two weeks. During this period, the collaborative and hybrid filtering approaches were tested. The data set consists of 130 ratings, assigned by 15 users on 110 books. Ratings have been given on a scale of 1 to 5, 1 being dreadful and 5 excellent.

The evaluation was done through three steps, as follows:

1. When a user requested recommendations, the system produced a list of recommended books, and for each book the system computed a predicted rating.

2. The system asked the user to assess the books, so the true user rating and the predicted rating were inserted into the database for evaluation purpose.
3. The system compared the true rating a user gave to the book with the prediction made by an algorithm, using the Mean Absolute Error algorithm.

6.2.3 Experimental Results

In this section, I present the results from applying different algorithms for generating recommendations. My results are mainly divided into two parts: the quality of results for each algorithm and the best performance algorithm. To test the quality of recommendations, I first determined the sensitivity of some parameters before running the experiment in the collaborative and hybrid filtering approaches. These parameters included (1) the minimum number of votes made by the active user to receive recommendations from the system, and (2) the number of common books between the active user and the others in the database neighbourhood for the active user.

The evaluation was done for the sensitivity of some parameters in collaborative and hybrid filtering approaches algorithms using off-line analysis. This was achieved by using the results of the live users' experiments from the database. The evaluation was done through three steps:

1. A list for each algorithm from the evaluation table in the database was created which contained the users' IDs, along with the evaluated book IDs and the users' ratings.
2. For each user in the created list from the previous step, the neighbour users' list was created using the proposed parameters for the evaluation. Then, the prediction rating for the evaluated book was computed using the neighbour users. This process was repeated for all the ratings of each user.
3. MAE was calculated using the true rating values and the prediction values for all the users.

6.2.3.1 Effect of Similarity Algorithms

I implemented two different similarities algorithms that were based on User-User Nearest Neighbourhood algorithm: the Pearson correlation coefficients and Mean Squared Differences

(MSD) algorithms. These two algorithms computed the degree of similarity between the active user and the others in the database. They were tested using the same dataset to show which one was more accurate for selecting neighbour users for the active user. I ran the experiment to compute the Mean Absolute Error (MAE) for each algorithm. **Figure 6.2** shows the results; using the Pearson coefficients algorithm as a similarity measure in the User-User Nearest Neighbourhood algorithm has clear advantages over the MAE algorithm. It is important to mention that the lower the Mean Absolute Error, the better performance can be classified.

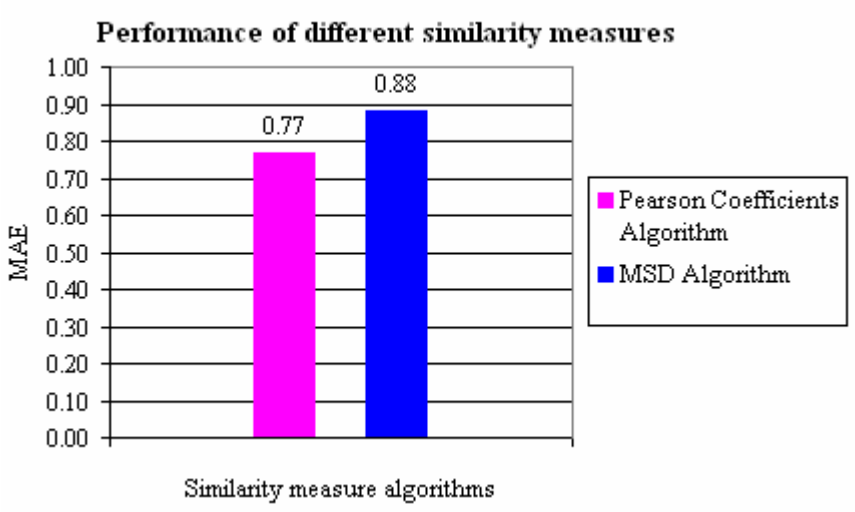


Figure 6.2: Effect of Similarity Measure Algorithms on Collaborative Filtering

6.2.3.2 Experiments with Number of Common Books

The number of common books between the active user and the other users in the database to be consider the neighbourhood for the active user, affects the accuracy of the recommendations. I performed an experiment to computes MAE for collaborative and hybrid filtering approaches with different numbers of common books between the active user and other users in the database, to be used to determine the sensitivity of this parameter. The results in **figure 6.3** show that the number of common books between the active user and the other users in the database has a big impact on the quality of the prediction rating. The results show that the collaborative filtering approach increased in prediction quality with an increase in the number of common books between the active user and other users in the database. On the other hand, we can observe in the hybrid approach that, as we increase the number of common books from 5 to 10, the prediction quality increases, and after that the curve tends to be rise.

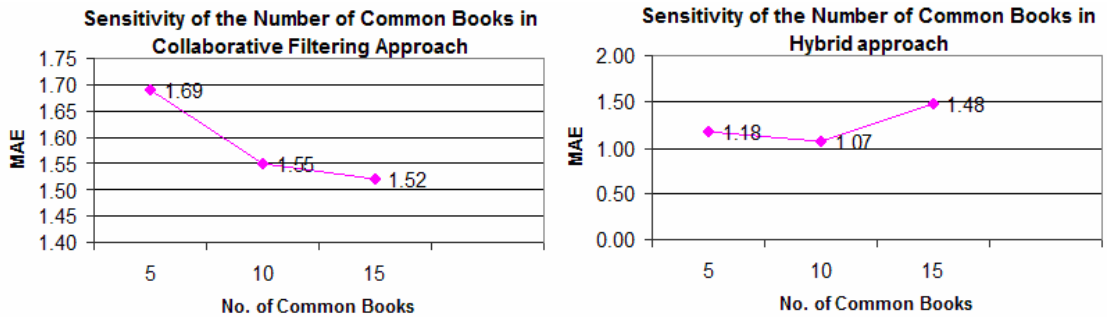


Figure 6.3: Sensitivity of the Number of Common Books

6.2.3.3 Experiments with Number of Votes by the Active User

The number of votes by the active user affects the accuracy of the recommendations. The more users vote, the better recommendations can be classified. I performed an offline analysis for three groups of users: users who had voted for between 10 to 15 books, those who voted for 15 to 20 books, and those who voted for more than 20 books. The results are shown in **figure 6.4.** These results were not expected, as we can observe a significant decreased in the prediction quality with increased number of votes by the active user. In my opinion, the reason behind the results is that the majority of users had assessed between 10 to 15 books.

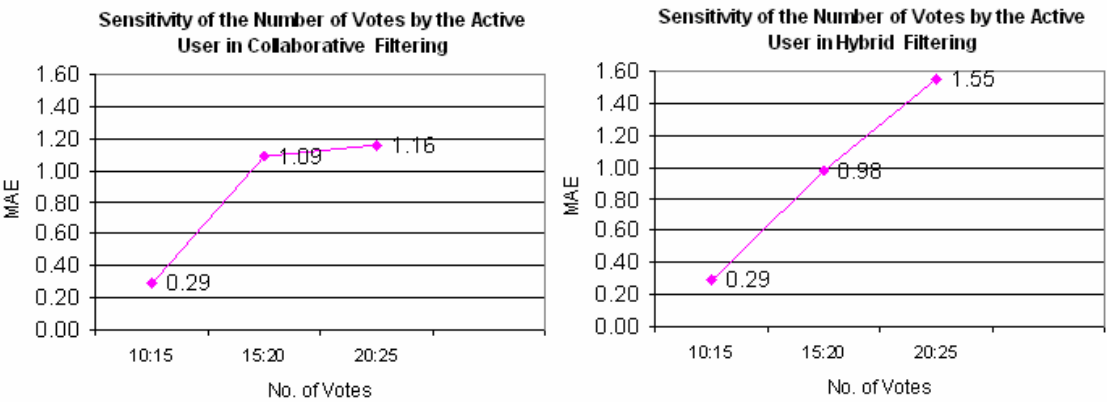


Figure 6.4: Sensitivity of the Number of Votes by the Active User

6.2.3.4 The Best Approach Performance

The content-based filtering technique was not evaluated due to time constraints. It uses a different accuracy matrix than the collaborative and hybrid approaches, so learning a new evaluation matrix will take a long time. In addition, the lack of standard content-bead recommender tests will make the comparison with the different recommender system

approaches difficult (Bogers and Bosch, 2007). One of the evaluation matrixes that content-based filtering used is the Mean Average Precision (MAP), which is based on a binary evaluation given by the users (Bogers and Bosch, 2007).

The experimental results show (**figure 6.5**) that the combination of content based and collaborative filtering, that is, the hybrid approach, has better performance than the collaborative filtering approach. The reason behind the success of the hybrid approach is that it uses the strength of one filtering technique to overcome the limitation of the other in order to provide accurate recommendations. Moreover, the content based approach is over-specialised because it is restricted to the user profile; while the collaborative filtering uses other similar users' experience. In contrast, collaborative filtering does not take into consideration the user's interests when producing the prediction rating for the recommended book.

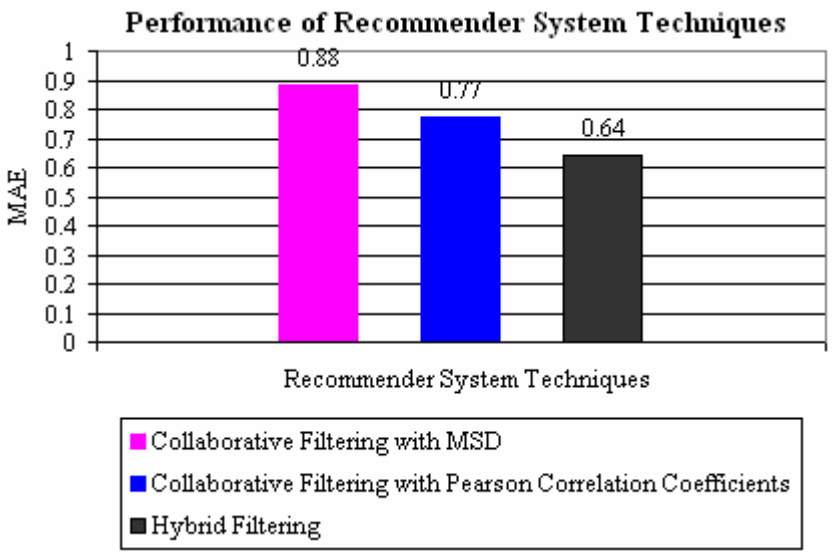


Figure 6.5 : Performance of Recommender System Techniques

To conclude, the evaluation results prove that the system recommendations are accurate. The system could be extended in the future to generate recommendations for other domains.

7 Conclusion & Project Management

This chapter presents conclusions about the project by first analyses the achievement of the project's objectives and then outlines the management of the project. Finally, possible future work is presented.

7.1 Analysis of Objectives

The objectives of the project were delivered at a high standard, with a few weaknesses occurring in the delivery of the second objective. The objectives will now be analyzed, based on those laid out in section 1.4.2.

The first objective required that research be conducted in an existing recommender system and profiling techniques. Chapter 2 had covered enough research into these two areas. The time and effort put into researching filtering and profiling techniques definitely was a good investment.

The second objective required the system to build a user profile that would represent the interests of the user. One of the important features of the user profile is that it keeps the interest of the user up to date. The user profile was to be built based on the categories and words of the books evaluated by the user. Due to my own experience this was achieved, but not to the standard to which I had planned. Data mining techniques were used for building profiles without using a stemming algorithm that would have streamed a word into its root. This requirement had to be omitted from the final implementation due to time constraints and the complexity of the project.

Objective three stated that the system should be built based on different approaches for computing recommendations. The project has developed a successful system which produces both non-personalised and personalised recommendations. Three different filtering approaches have been implemented for producing personalised recommendations. First, a content-based technique produces recommendations using the active user's profile which contains evaluations provided by the user. Second, a collaborative filtering technique produces recommendations based on the active user evaluations and other similar users. Finally, a hybrid approach produces recommendations by combining the content-based and collaborative filtering approaches. Collaborative and hybrid filtering use the same algorithm,

except that the hybrid approach includes the categories in the active user's profile to add an extra value for the recommended book's prediction.

The last objective required the project to evaluate the accuracy of the system recommendations. It is disappointing that the content-based filtering technique was not evaluated, but the project objectives were too ambitious. When the objectives were set, the size and complexity of the project were not understood and so were underestimated. The content-based filtering approach uses a different accuracy matrix than the collaborative and hybrid approaches, so learning a new evaluation matrix will take a long time.

The evolution of the implemented algorithms was the hardest task in the project. It took a considerable amount of research and experimentation to examine and evaluate the behaviour of the proposed system at a high standard. The experiments were done for the three different algorithms (collaborative filtering with person correlation coefficient algorithm, collaborative filtering with MSD algorithm and hybrid approach). The evaluation results show that using the Pearson coefficient for the similarity measure in the collaborative filtering approach is more accurate in selecting neighbour users for the active user than using the Mean Squared Differences algorithm. In addition, a recommender system based on the combination of content-based and collaborative filtering approaches provides more accurate recommendations for the book domain.

As can be seen from this analysis, the project objectives have been completed at a satisfactory level. Also, we can see from the system evaluation results that a recommender system is an efficient approach to solve the problem of information overload in e-commercial web sites.

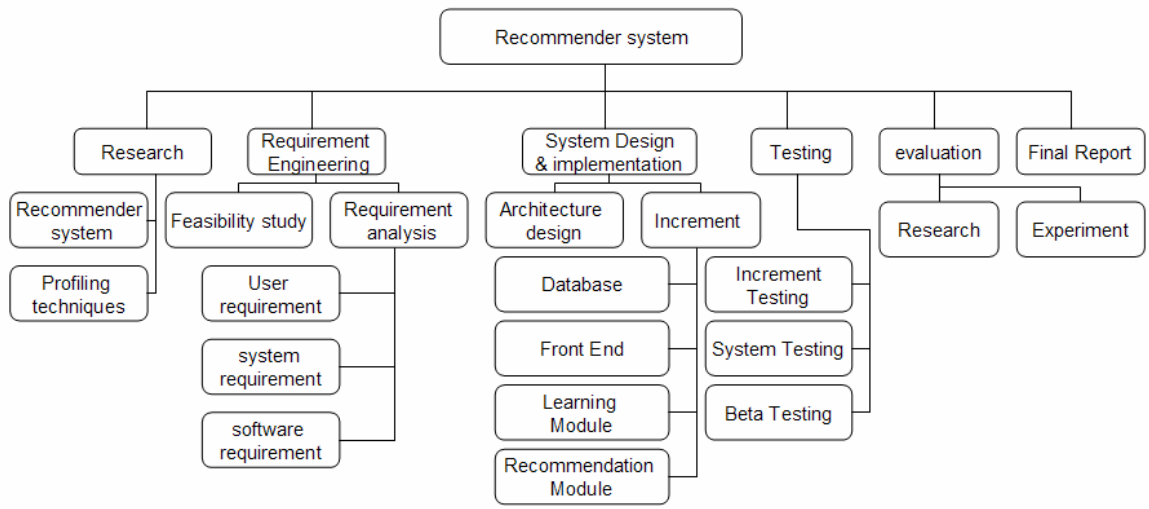
7.2 Project Management

Time planning is important, especially in a case like this, which involves such strict time constraints. To make a project a success, setting deadlines for completion tasks is important. In fact, I started working on this project with the CE902 course, for which I was responsible for submitting a proposal for my graduate project. I started at Christmas vacation by doing some background reading in the area of agent technology for e-commerce especially in the field of recommender systems, which is the focus of my dissertation. In addition, meetings and discussions with my supervisor Dr.Fasli were held regularly to make sure that the project

was on the right track; these gave me the opportunity to gather valuable comments and suggestions from my supervisor.

7.2.1 Work Plan

The project was developed using an incremental approach to delivery, and this led to the successful production of the project. The tasks I needed to undertake to develop the system were identified using a Work Breakdown Structure method; these tasks are shown below:



This was then used to create a time plan. This time plan was used to create a Gantt chart to monitor the project activity (see **Appendix M**). In addition, the plan included some slack time in case the project started to run behind schedule. The Gantt chart was very useful in the planning aspects of the project, as it allowed all of the tasks and their subtasks to be displayed within the desired timeframe.

7.2.2 Evaluation

The initial project plan that I have used to manage my project has really helped me achieved the project at a satisfactory level. Although the time planning of each task was accurate, some of the tasks were either underestimated or overestimated. Due to lack experience with project planning and the project field, the project started to fall behind schedule from early on. Having recognized this, I was able to avert a potential delay occurring in my project and adjust my project plan. It was very important to make this adjustment, to ensure the project remained on schedule.

The time required for searching and investigating recommender system and profiling techniques was underestimated, as it took much longer than planned, throughout the entire duration of the project. This was the main reason for the slow the progress through the early stages. As I had no previous knowledge of the intelligent agent area, the searching and learning was very slow. However, the result was invaluable, as it aided me in more reasonably estimating the project scope and in understanding clearly the techniques that helped me towards getting the project back on schedule.

The design stage of this project was managed very successfully. It took a lot of effort because a well designed application with a minimum amount of duplicated code can be further improved. Much of the implementation time was spent on developing the collaborative filtering approach, which took an extra week to complete. However, the extra time needed for this phase was made up, as the development of content based and hybrid approaches saved me a lot of time, which resulted in the flexibility in the plan.

The evaluation of the system recommendations was the hardest task in the project, but it was managed as expected. Finally, writing the report was achieved. Every section of the document is written as soon as the associated work is completed to ensure that the final report will completed early. The reason behind this is that I planned to send the report for proof reading in order to make sure the document would be free of grammatical errors, as English is not my first language.

To conclude, it is important to realize that project management skills play an important role in the success of the system. If the project could be completed again, more time would be allocated for the investigation stage, as learning a new thing requires more time. The process of completing the project provided me with the experience that will be invaluable to me in my future work.

7.3 Personal Thoughts and Experience

Information filtering and profiling techniques are important fields today in commercial websites. This fact accounts for my interest in conducting research project in the area of e-commerce Intelligent Agents. In addition, I thought this project type would be an excellent opportunity to showcase the new skills I have acquired.

A major difficulty faced was that I am unfamiliar with recommender system and data mining techniques. It took a lot of time and effort in researching and experimentation, but this was a good investment. Although the experiments were the most challenging tasks in the project, they were also the most exciting, as they proved that the proposed system works as I had imagined. Whilst developing the web application for this project I have become more familiar with .NET technology. My understanding of C# and Object Oriented Programming has improved greatly during the project. I would now recommend .NET as a technology for web application development because it proved to be a very powerful and flexible framework.

To conclude, I am very proud of the outcome of this project and look forward to working in the Intelligent Agent area in my future study.

7.4 Further Work

Although the objectives specified for the developed system have been achieved, there are some areas that can improve the system's functionality and usability. The following improvements are the most important:

- **Improve user's profile:** The user's profile has been developed using the categories and words of the books evaluated by the user. Further work can be done to make a significant improvement in profile accuracy by using demographic data profiles such as age and sex. In addition, the user's profile can be extended to take into consideration the search term provided by the user. Also, it would be beneficial to introduce more classification of books into the user profile, for example by author and publisher, instead of only book's categories.
- **Scalability issues in collaborative filtering:** The collaborative filtering algorithm should be adapted to address the scalability issue as the number of users increases and their profile size becomes large.
- **Extending learning module:** The learning module of the system can be extended by allowing the users to provide textual comments on the books, which provides additional information for other users to read. The system can filter and then record the textual comments provided by the user.
- **Other domains:** The system could be extended to contain other domains such as movies and songs.

- **Evaluate content-based filtering approach:** Future work can be done to evaluate the accuracy of the content-based filtering recommendations. Then, an investigation should be conducted to identify which existing evaluation matrix could evaluate the accuracy between different recommender system techniques.
- **Extending experiment:** The experiments were conducted on the system in a short period of time and with a small data set, due to time constraints. Conducting additional experiments over a longer period and with different data sets would verify the evaluation results; it would improve the recommendation accuracy. So, too, would adjusting some of the similarity measure algorithm parameters in collaborative and hybrid filtering approaches, such as the number of neighbour users for the active user. In addition, an interface for the experiment process could be provided, which would allow the evaluator to enter different values within different algorithms without changing any thing in the code.

8 References and Bibliography

8.1 References

Breese, J., Heckerman, D. and Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence* (UAI-98), pages 43-52, Madison, WI.

Bogers, T. and Bosch, A. (2007).Comparing and Evaluating Information Retrieval Algorithms for News Recommendation. Tilburg University. Minneapolis, Minnesota, USA.

Dunham, H. (2003). Data Mining: Introductory and Advanced Topics. Pearson Education, Inc. (Prentice Hall), ISBN 0130888923.

Fasli, M. (2006). Agent Technology for E-Commerce. Chichester: John Wiley, ISBN 0470030305, pages 149-165.

Fawcett, J. and Valenzuela, S. (2008). Professional ASP.NET Design Patterns. Wrox , ISBN 0470292784.

Gauch, S., Speretta, M., Chandramouli, A. And Micarelli, A. (2007).User profiles for personalized information access. In Brusilovsky, P., Kobsa, A. And Nejdl, W. (eds.): The AdaptiveWeb: Methods and Strategies of Web Personalization. Lecture Notes in Computer Science, Vol. 4321. Springer, Berlin.

Goldberg, D., Nichols, D., Oki, B and Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12): pages 61-70.

Hayes, C., Massa, P., Avesani, P. and Cunningham P. (2002). An online evaluation framework for recommender systems. In *Proceedings of the Workshop on Personalization and Recommendation in E-Commerce (RPEC).* Springer-Verlag, Malaga, Spain.

Herlocker, J., Konstan, J., Terveen, L. and Riedl, J. (2004). Evaluating Collaborative Filtering Recommender Systems. ACM Transactions on Information Systems, 22(1): pages 5-53.

Hung, M. and Zou, Y. (2005). Extracting Business Policies and Business Data from the Three-Tier Architecture Systems. In *Proceedings of International Workshop on Reverse Engineering to Requirements*, pages 24-28, Pittsburgh, Pennsylvania.

Jennings, N., Moreau, L. and Wei, Y. (2005). A Market-Based Approach to Recommender Systems. Intelligence Agents Multimedia Group, School of Electronics and Computer Science, University of Southampton, Southampton, U.K.

Karypis, G. (2001). Evaluation of Item-Based Top-N recommendation Algorithms. In *proceedings of the 10th Conference of Information and Knowledge Management (CIKM)*, pages 247-254.

Konstan, J. A., Schafer, J. B. and Riedl, J. and (2001). E-commerce recommendation applications. *Data Mining and Knowledge Discovery*, 5(1/2): pages 115-153.

Lang, K. (1995). NewsWeeder: learning to filter netnews. In *Proceedings of the 12th International Conference on Machine Learning (ICML'95)*, pages 331-339, Tahoe city, CA.

Linden, G., Smith, B. and York, J. (2003). Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1): pages 76 – 80.

Losee, R. (1989). Minimizing information overload: The ranking of electronic messages. Journal of Information Science, 15(3): pages 179-189.

Middleton, S. (2003). Capturing knowledge of user preferences with recommender systems. PhD thesis, University of Southampton.

Mortensen, M. (2007). Designing and Evaluation of a recommender system. Department of computer science. MSc thesis, University of Tromso.

Pazzani, M. (1999). A framework for collaborative, content-based and demographic filtering. Artificial Intelligence Review, 13(5-6): pages 393-408.

Rashid, A., Karypis, G. and Riedl J. (2005). Influence in Ratings-Based Recommender Systems: An Algorithm- Independent Approach. *SIAM International conference on Data Mining.*

Resnick, P. and Varian, H. (1997). Recommender Systems. *Communications of the ACM*, 40(3): pages 56-58.

Sarwar, B., Karypis, G., Konstan, J. and Reidl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International World Wide Web Conference*, pages 285-295, Hong Kong, China.

Shardanand, U. and Maes, P. (1995). Social information filtering: Algorithms for automating "word of mouth". In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI'95), pages 210-217, Denver, CO.

Sommerville, I. (2006). Software Engineering. Pearson Education, ISBN 0321313798, pages 392-394.

Turban, E., Lee, J., King, D. and Chung, H. (2000). Electronic Commerce: A Managerial Perspective. Prentice Hall, Englewood, Cliffs, NJ.

8.2 Web References

Amazon.com
Last accessed:16 August 2008, from
http://www.amazon.com/

Dundas Tools
Last accessed:16 August 2008, from
http://www.dundas.com/Products/Chart/NET/index.aspx

Group Lens Research Projects
Last accessed: 1 August 2008, from
http://www.grouplens.org

Movie Lens
Last accessed: 20 August 2008, from
http://www.movielens.org

MSDN.microsoft.com
Last accessed: 16 August 2008, from
http://msdn2.microsoft.com/en-us/library/aa139615.aspx

The Source for Java Developers
Last accessed: 16 August 2008, from
http://www.sun.java.com

TIOBE Programming Community Index for November
Last accessed: 16 August 2008, from
http://www.tiobe.com/tpci.htm

8.3 Bibliography

Ansari, A., Essegaier, S. and Kohli, R. (2000). Internet Recommendation System. Journal of Marketing Research (37:3), pages 363-375.

Bloedorn, E., Mani, I. and MacMillan, T. (1996). Machine Learning of User Profiles: Representational Issues. In *Proceedings of AAAI 96*, pages 433-438, Portland, Oregon.

Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D. and Sartin, M. (1999). Combining content-based and collaborative filters in an online newspaper. In *Proceedings of the ACMSIGIR Workshop on Recommender Systems*, Berkeley, CA.

Crabtree, B. and Soltysiak, S. (1998). Identifying and Tracking Changing Interests. International Journal on Digital Libraries, 2(1), pages 38-53.

Good, N., Schafer, B., Konstan, J., Borchers, A., Sarwar, B., Herlocker, J., and Riedl, J. (1999). Combining Collaborative Filtering With Personal Agents for Better Recommendations. In *Proceedings of the AAAI-'99 conference*, pages 439-446.

Hetzel, W. (1998). The Complete Guide to Software Testing, 2nd ed. Wellesley , Mass.

Herlocker, J., Konstan, J., Borchers, A. and Riedl, J.(1999). An algorithmic framework for performing collaborative filtering. In *proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99),* pages 230 – 237. Berkeley, CA.

Herlocker, J., Konstan, J. and Riedl, J. (2000). Explaining collaborative filtering recommendations. In *Proceedings of ACM Conference on Computer Supported Cooperative Work*, pages 241-250, Philadelphia, PA.

Montaner, M., Lopez, B. and Dela, J. (2003). A taxonomy of recommender agents on the internet. Artificial Intelligence Review, pages 285–330.

Rashid, A., Albert, I., Cosley, D., Lam, S., McNee, S. , Konstan, J. and Riedl, J (2002). Getting to Know You: Learning New User Preferences in Recommender Systems. In *Proceedings of the 2002 International Conference on Intelligent User Interfaces*, pages 127-134, San Francisco, CA.

Resnick, P., Lacovou, N., Suchak, M., Bergstorm, P. and Riedl, J. (1994). Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, pages 175-186 ,Chapel Hill, NC.

Sheth, B. and Maes, P. (1993). Evolving agents for personalized information filtering. In Proceedings of the Ninth Conference on Artificial Intelligence for Applications (CAIA'93), pages 345 – 350, Orlando, FL.

Wasfi, A. (1999). Collecting User Access Patterns for Building User Profiles and Collaborative Filtering. In *Proceedings of the 1999 International Conference on Intelligent User Interfaces*, pages 57-64.

9 Appendices

9.1 Appendix A: Recommender Systems in E-Commerce

Recommender systems help e-commerce sites to increase their sales through suggesting products or services to their users. These recommendations based on the top overall sellers on a site, the demographics of the user, or the analysis of the past buying behaviour of the user. In addition, recommender systems enhance e-commerce sites through three ways (Konstan et al., 1999):

- **Browsers into consumers:** visitors of a website often browse the website without thinking of purchasing anything. Recommender systems can help consumers to find items they might be interested in purchasing them through recommendations that act as advertisements (Fasli, 2007).
- **Cross-sell:** the average order size for the consumer should be increased if the recommendations given by the recommender system are good. For instance, a site might recommend additional products or services in the checkout process, based on those products in the consumer shopping cart.
- **Loyalty:** Recommender systems improve loyalty by creating a relationship between the site and the consumer. It helps sites learning about their consumers according to the consumer's preferences. Also, recommender systems present custom interface that matches the consumer needs. In deed, the more consumers use the recommendation because of the matching of their interests the more loyal they are to the site.

9.2 Appendix B: Providing Recommendations

Different forms for providing recommendations have been developed, however they can be classified into the following forms:

- **Attribute-based recommendations**

The recommendations generated by the system are based on syntactic properties of the products. Consumers must directly request the recommendations by entering his/her desired syntactic product properties. For instance, when a consumer searches for a computing book and the recommender system responds with a list of computing books. In addition, the recommender system can make recommendations correlate with the user's preferences kept in profile (Fasli, 2006).

- **Item-to-Item Correlation**

Item-to-item recommendations are based on a set of items in which the user has expressed previously interest in. For example, if a user has placed some products in her/his shopping basket, the recommender system may recommend complementary products to increase the order size. The item-to-item recommendations based on the user behaviour and are generated form the currently selected products by the user.

- **People-to-people correlation**

People-to-people recommendations are based on the correlations between the active user seeking recommendation and other users who have expressed preferences or purchased products (Fasli, 2006).

- **Non-personalised recommendations**

Non-personalised recommendations are based on the average rating of people opinion about a specific item. The recommendations are independent of the user which required less effort to generate the recommendations. These kinds of recommendations are common in physical stores, since they are not personalised for each consumer.

9.3 Appendix C: Use Cases

Accounts Package:

Use Case	Register
Description	A guest should be encouraged to register on the web site in order to take advantage of benefits that are only open to members. The registration process will capture personal details, knowledge based on the user's preferences and login information.
Actors	The guest
Pre-conditions	The guest is a non-member.
Post-conditions	The guest becomes a member.
Main Flows	1. The user requests a new account with the system using the Registration Page Form. 2. The user provides login information and personal as well as preferences details. 3. Validation will be applied to these fields to minimize the insertion of erroneous data. 4. Passwords will be encrypted within the database for security purposes. 5. The system registers the user, and redirects the user to his/her control panel.
Alternative Flows	If the user cancels his/her registration, the form will be cleared. If the user doesn't fill in all the required fields and presses 'submit' button, an error message will appear. If the user enters a field value which doesn't comply with a certain validation rule, an error message will appear and the system will offer the user a chance to enter a valid data. If the user enters an email address or/and a username already registered with the system, the system will display an error massage and offer the user a chance.

Use Case	Login
Description	A login page will provide an interface to authenticate a user before using restricted features.
Actors	The user
Pre-conditions	The user is a member.
Post-conditions	The user is logged into the system and will remain logged in until he/she loges out or leaves the website for an extended period of time.
Main Flows	1. The user requests to log in with the system using the Login Page Form. 2. The login form requires the capture of a username and password. 3. The user enters a username and password. 4. The system verifies the username and password. 5. The system signs the user into the website, and redirects the user to his/her Control Panel.
Alternative Flows	If the user is not registered, the system will request the user to register instead of login. If the user enters an incorrect username and/or password, the system will display an error message and offer another chance to enter the correct account details.

Use Case	Edit Profile
Description	A user may edit his/her profile to change his/her personal or preference information.
Actors	The user
Pre-conditions	The profile page will appear if the user loges in.
Post-conditions	The user's profile has been modified.
Main Flows	1. The user requests to edit their profile. 2. The user enters the new information. 3. The system verifies and saves the changes.
Alternative Flows	If the user enters an incorrect personal data, the system will display an error message to inform the user.

Use Case	Statistical
Description	A statistical chart which represents a user's interest in a certain category.
Actors	The user
Pre-conditions	The statistical page will appear if the user loges in.
Post-conditions	The statistical chart has been showed.
Main Flows	1. The user selects a certain category. 2. The system shows a statistical graph based on the selected category.
Alternative Flows	If the user doesn't select a category, the system will display an error message to inform the user. If The selected category does not contain any data about the user's interest, the system will display a message to inform the user and encourage him/her to evaluate more books.

Use Case	View Shopping cart
Description	A user should be able to view his/her shopping cart.
Actors	The user
Pre-conditions	The user must be logged in.
Post-conditions	The shopping cart has been showed.
Main Flows	1. The user requests to view his/her shopping cart. 2. The system shows books in the user's shopping cart.
Alternative Flows	If there are no books in the user's shopping cart, the system will display a message to inform the user.

Browsing Package:

Use Case	Search
Description	There are two types of search: basic and advanced. Any visitor to the site can conduct a search.
Actors	The user
Pre-conditions	The user writes the keyword or fills out advanced search form for books that he/she is looking for.
Post-conditions	The search results appeared.
Main Flows	1. The user enters a keyword before submitting the search form. 2. The system checks for matching entries and displays the results to the user. 3. The user selects the book she/he wishes to view.
Alternative Flows	1. If the user wishes to perform an advanced search, Step 1 will be replaced with a form to the user; the user fills out and submits the advanced search form. 2. If there are no matches to the search term, the system will inform the user.

Use Case	Search by Author Name
Description	The automatic creation of hyperlinked authors should feature on book's details. This will enable one book to be related to another.
Actors	The user
Pre-conditions	The user presses an author name hyper link.
Post-conditions	The search results appeared.
Main Flow	1. The user views a certain book details and presses an author name hyperlink. 2. The system checks all the books written by the author, and then displays the results to the user. 3. The user selects the book she/he wishes to view.
Alternative flows	-

Use Case	Browse Category
Description	A user browses the system to view a certain category.
Actors	The user
Pre-conditions	-
Post-conditions	The search results appeared.
Main Flow	1. The user selects a category. 2. The user selects a sub-category. [optional] 3. The system displays a set of books within the selected category.

Use Case	View Book Details
Description	A user may browse the system or conduct a search to view a book. This extends the Search or Browse Category functionality since the user selects a book from the results.
Actors	The user
Pre-conditions	The user conducts a search or browses for a certain book.
Post-conditions	The user views the book details.
Main Flows	[Extend: Search or Browse Category] 1. The user selects a book. 2. The system returns the requested book details, and displays a predicted rate for the book using the Mean algorithm along with the number of the users who rate the book as a non-personalised recommendation. 3. The system displays a statistical chart based on the users' rates as a non-personalised recommendation. 4. The system will display two kinds of a non-personalised recommendation based on what a customer bought and viewed along with this book. 5. If the user logged in, the system will show 'add to favourite' button, and add the book to the user browsing history and update the user's profile.
Alternative Flows	Step 4, if there are no users bought or viewed this book, the system won't show personalised recommendations.

Learning Module Package:

Use Case	Rating
Description	The member should be able to rate books in a numeric scale which will improve his/her recommendation.
Actors	The user
Pre-conditions	-.
Post-conditions	The system saves the user's rate.
Main Flow	[Extend: View Book Details] 1. The user selects a rate from 1 to 5 numeric scales, and presses the submit button. 2. The system saves the user's rate, and updates the user's profile.
Alternative flows	If the user is not logged in, the system will redirect the user to the login page step 1 [Include: Login]. If the user presses submit button without selecting a rate, the system will display an error message and offer the user a chance to select a rate. If the user had rated the book previously, the system would update the user's rate.

Use Case	Add to Favourite
Description	The member should be able to add a specific book to their favourite book list.
Actors	The user
Pre-conditions	-
Post-conditions	The system adds the selected book to the user's favourite book list.
Main Flows	[Extend: View Book Details] 1. The user presses the Add to the favourite button. 2. The system adds the book to the user's favourite book list, and updates the user's profile.
Alternative Flows	If the user is not logged in, the system will redirect the user to the Login page, step 1 [Include: Login]. If the user had added the book previously, the system will only update the user's profile.

Use Case	Add to Shopping Cart
Description	The member should be able to add a specific book to his/her shopping cart.
Actors	The user
Pre-conditions	-
Post-conditions	The system adds the selected book to the user's shopping cart.
Main Flows	[Extend: View Book Details] 1. The user presses the Add to cart button. 2. The system adds the book to the user's shopping cart, and updates the user's profile.
Alternative Flows	If the user is not logged in, the system will redirect the user to the login page, step 1 [Include: Login]. If the book is already in the user's shopping cart, the system will increment the book quantity by one.

Use Case	Add to Owned Book
Description	The member should encourage marking any book, that she/he owns, in order to improve its recommendations.
Actors	The user
Pre-conditions	-
Post-conditions	The system adds the marked book to the user's owned book list.
Main Flow	[Extend: View Book Details] 1. The user marks *I owned this book* check box. 2. The system adds the book to the user's owned book list, and updates the user's profile.
Alternative flows	If the user is not logged in, the system will redirect the user to the login page, step 1 [Include: Login]. If the book is already in the user's owned book list, the system will only update the user's profile.

Improve Recommendation Package:

Use Case	View Rated book
Description	The user should be able to view his/her rated books to improve his/her recommendations.
Actors	The user
Pre-conditions	When the user is logged in, the improved recommendation page will appear.
Post-conditions	The user is on his Rated book page viewing his rated books.
Main Flow	1. The user presses View Rated Book link. 2. The system returns a list of rated books selected by the user. 3. [optional] the user selects a book to delete it by pressing the Delete link(no more interesting in this book].[recursive] 4. The system saves the changes, updates the user's profile and redirects the user to the same page.
Alternative flows	If the user has no rated book, the system will display a message to inform the user.

Use Case	View User's Favourite Book
Description	The user should be able to view his/her favourite books.
Actors	The user
Pre-conditions	When the user is logged in, the improved recommendation page appears.
Post-conditions	The user is viewing a list of his/her favourite books.
Main Flow	1. The user presses View Favourite Book link. 2. The system returns a list of the user's favourite books. 3. If the user is not interesting in a book, he/she can delete it by pressing the delete link. 4. The system saves the changes, updates the user's profile and redirects him/her to the same page.
Alternative flows	If the user has no favourite book, the system will display a message to inform the user.

Use Case	View User's Owned Book
Description	The user should be able to view a list of books he/she owns.
Actors	The user
Pre-conditions	When the user is logged in, the improved recommendation page appears.
Post-conditions	The user is viewing a list of book s/he own.
Main Flows	1. The user presses Owned Book link. 2. The system returns a list of books that the user owns. 3. If the user is not interesting in a book, he/she can delete it by pressing the delete link. 4. The system saves the changes, updates the user's profile and redirects the user to the same page.
Alternative flows	If the user has no owned book, the system will display a message to inform the user.

Use Case	View User's Browsing History
Description	The user should be able to view his/her browsing history in order to improve his/her recommendation.
Actors	The user
Pre-conditions	When the user is logged in, the improved recommendation page appears.
Post-conditions	The user is viewing a list of his recently browsed books.
Main Flows	1. The user presses Browsing History link. 2. The system returns a list of the recently viewed books by the user. 3. If the user is not interesting in a book, he/she can delete it by pressing the delete link. 4. The system saves the changes, updates the user's profile and redirects him/her to the same page.
Alternative Flows	If the user has no book in their browsing history; the system will display a message to inform the user.

Recommendation Module Package:

Use Case	Personalised Recommendation
Description	The users should be able to ask for personalised recommendations based on their profiles and/or similar users' preferences. These recommendations must be produced using different algorithms in order to evaluate them during the project experiment.
Actors	The user
Pre-conditions	The user is logged in.
Post-conditions	The user receives a list of recommend books based on his/her requests.
Main Flows	1. The user asks for recommendations based on a certain approach (content based, collaborative filtering approach using MSD for similarity measurement, collaborative filtering approach using Pearson correlation for similarity measurement, or the hybrid approach). 2. The system returns a list of recommend books to the user.
Alternative Flows	The user has no recommendations because no similar users are founded and/or the user profile does not contain enough data about the user interest. The system will display a message to inform the user and ask him/her to evaluate more books in order to receive recommendations.

Use Case	Non-Personalised Recommendation
Description	Any user should be able to receive non-personalised recommendations based on the books bestsellers, and the most recently viewed books in this month according to the users' browsing history.
Actors	The guest/user
Pre-conditions	-
Post-conditions	The user receives a list of recommend book based on his/her requests.
Main Flows	1. The user asks for non-personalised recommendations. 2. The system returns a list of recommend books to the user.
Alternative Flows	-

9.4 Appendix D: Network Activity Diagrams

From each use case, an activity diagram has been prepared.

Account:

Register

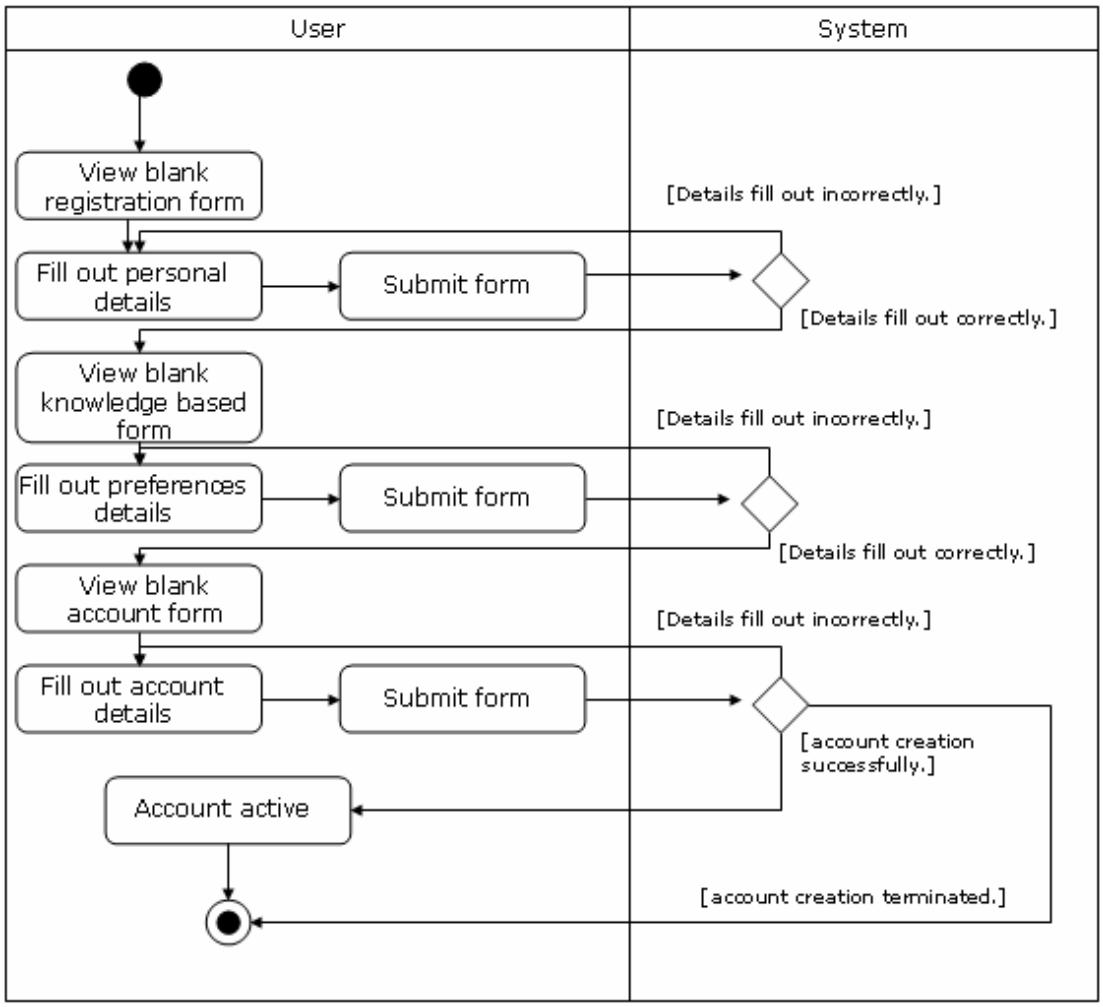


Figure 9.1: Activity Diagram to Model the User Registration Process

Login

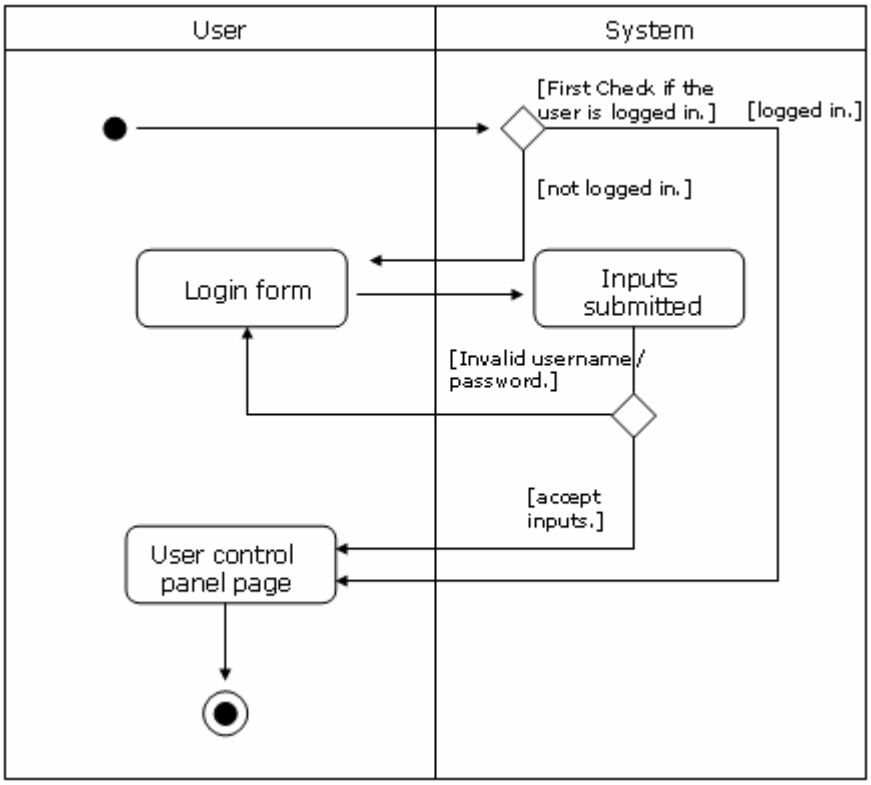


Figure 9.2: Activity Diagram to Model the User Login Process

Edit Profile

Edit personal details

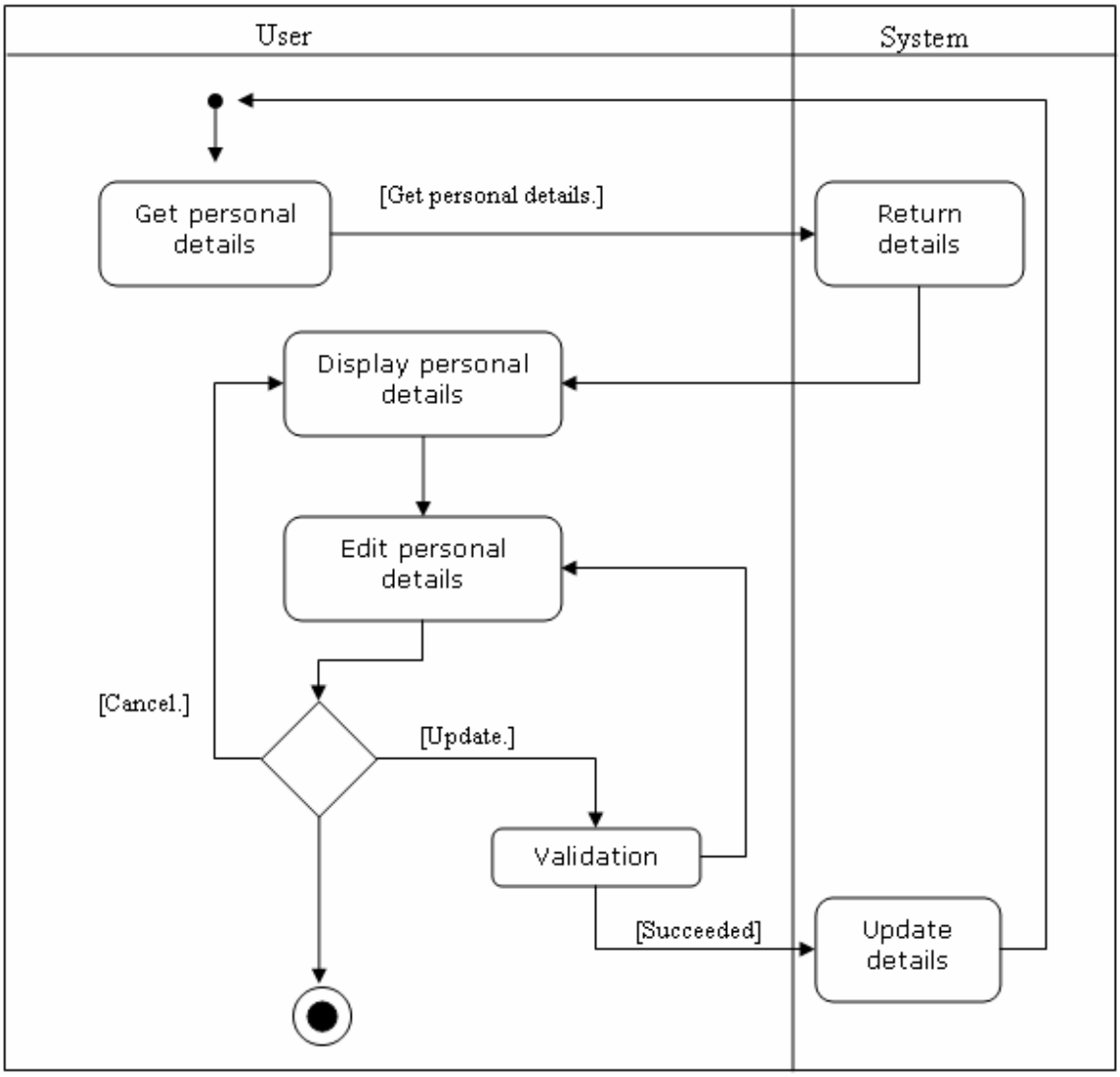


Figure 9.3: Activity Diagram to Model the Profile Editing process

Edit preferences details

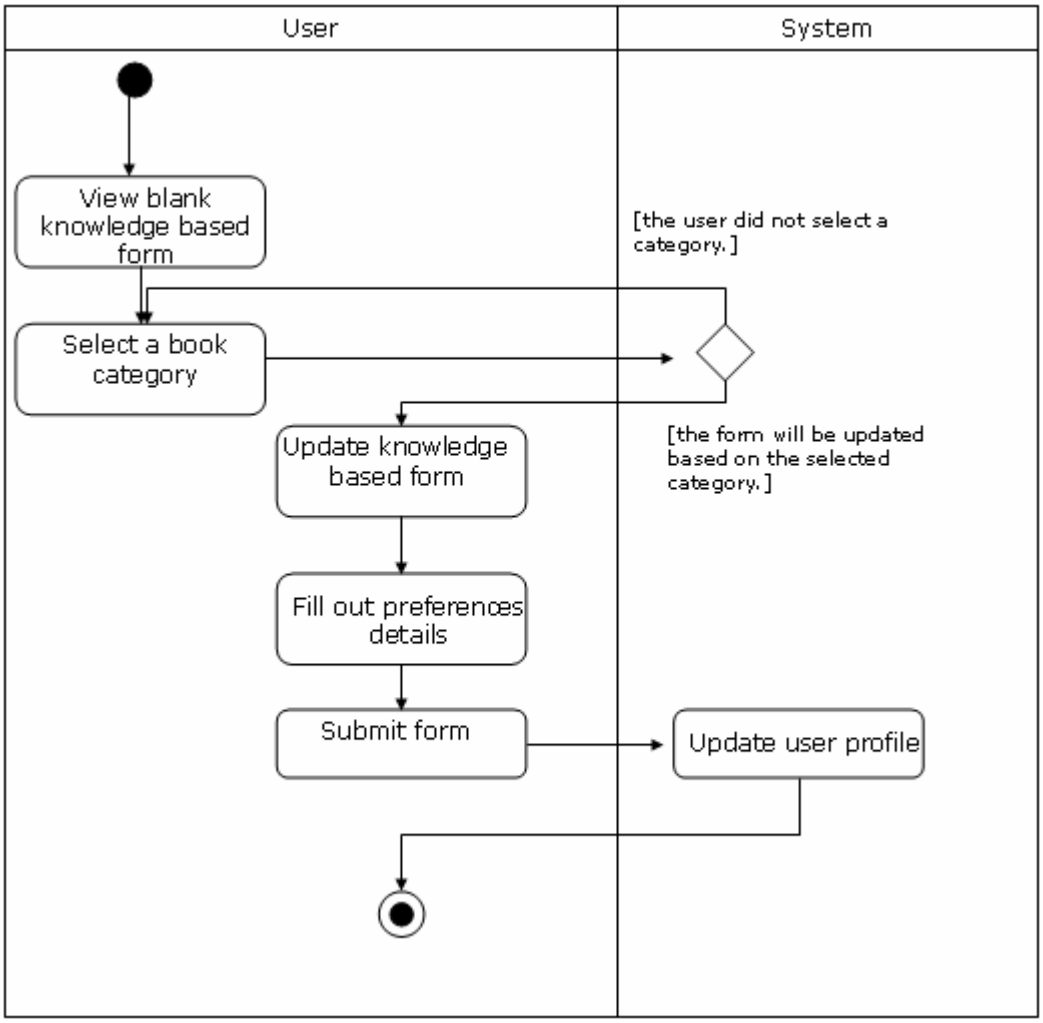


Figure 9.4: Activity Diagram to Model the Preference Elicitation Form process

User's Preferences Graph

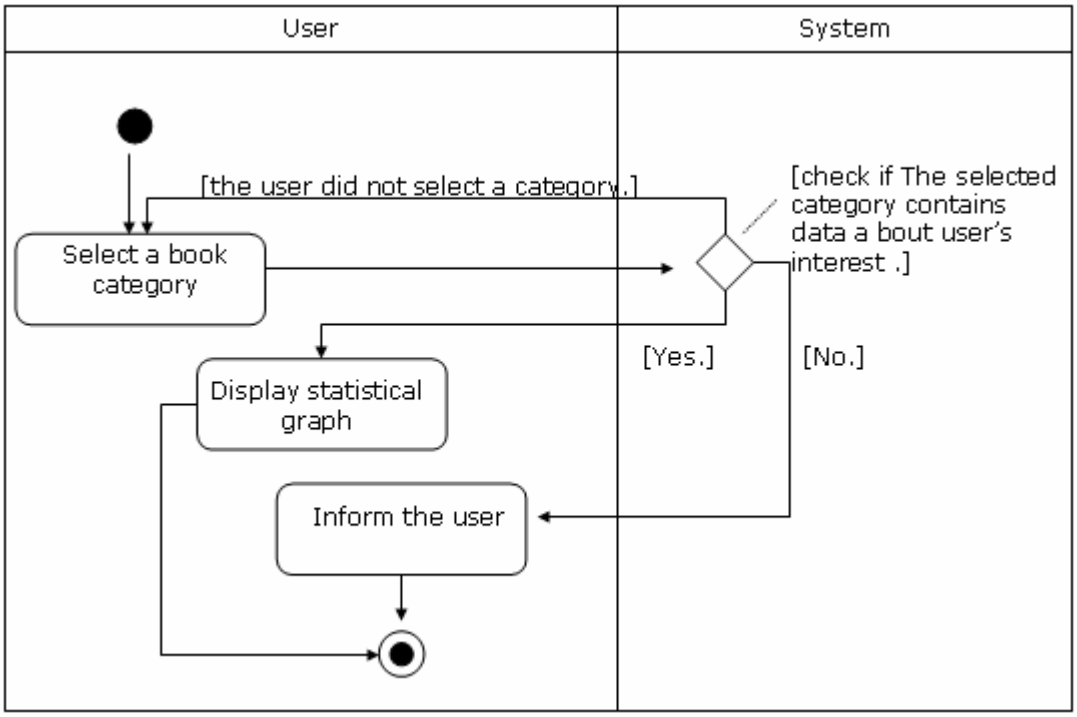


Figure 9.5: Activity Diagram - User's Preferences Graph

View Shopping Cart

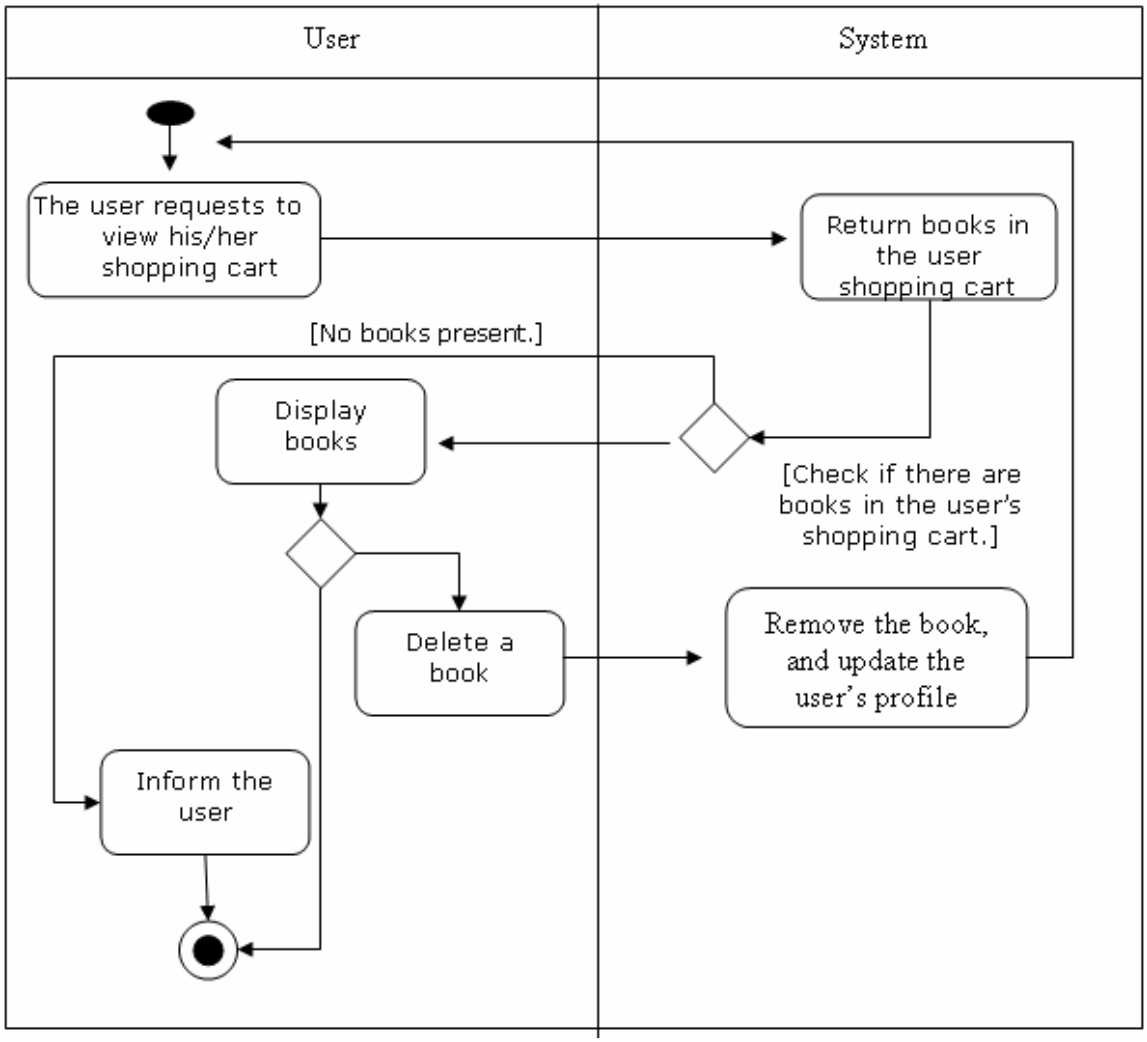


Figure 9.6: Activity Diagram - View Shopping Cart

Search

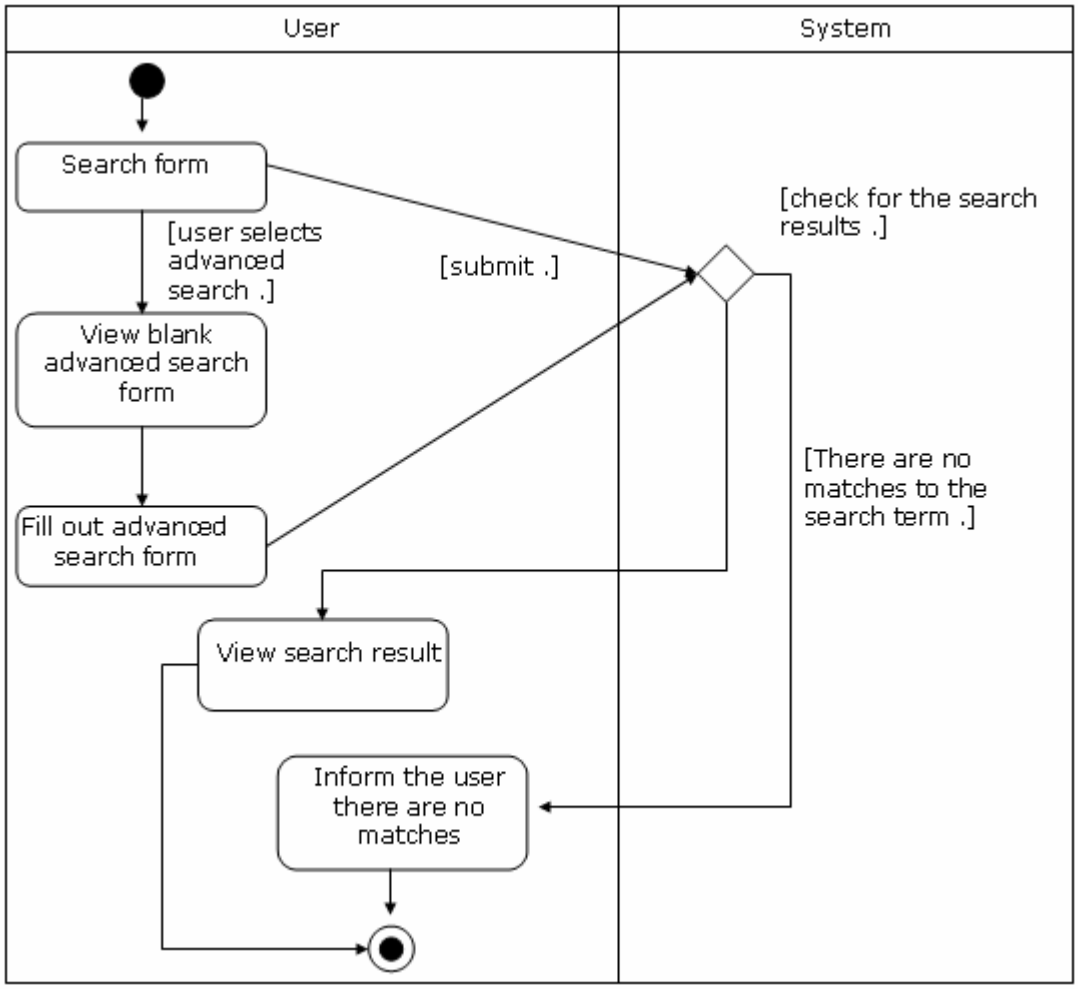


Figure 9.7: Activity Diagram - Search

Search by Author Name

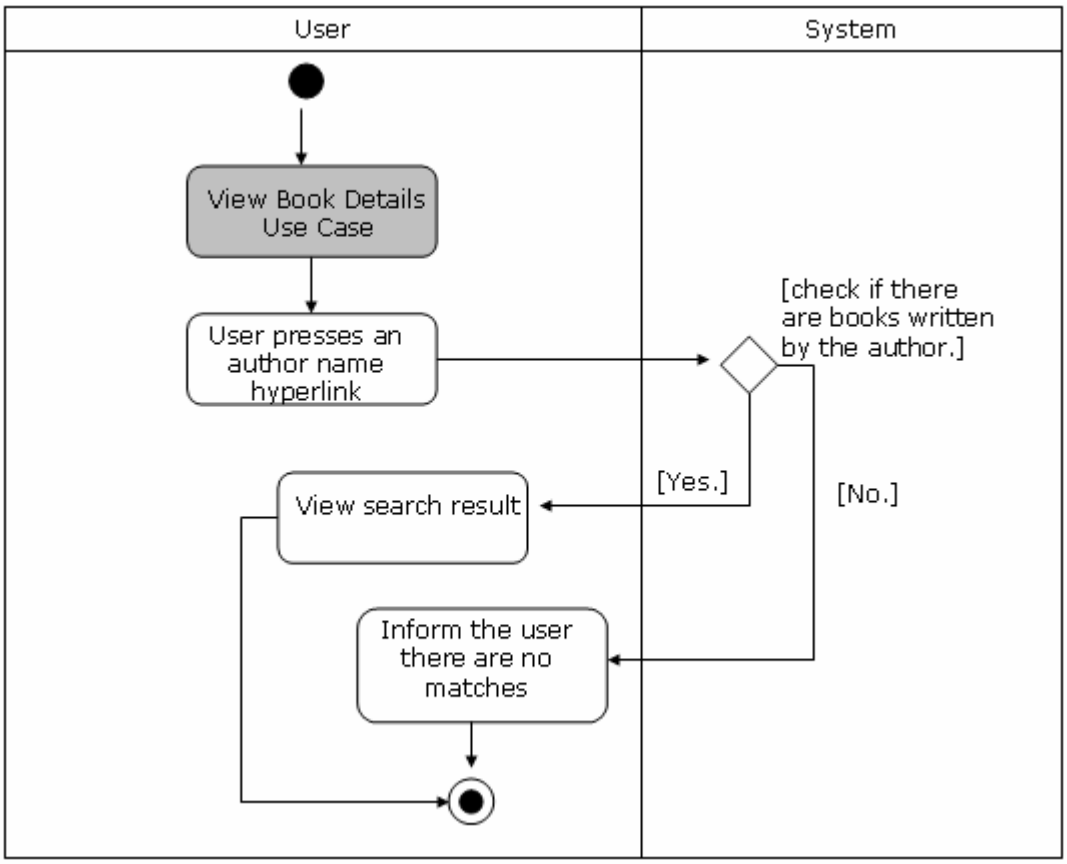


Figure 9.8: Activity Diagram - Search by Author Name

Browse Category

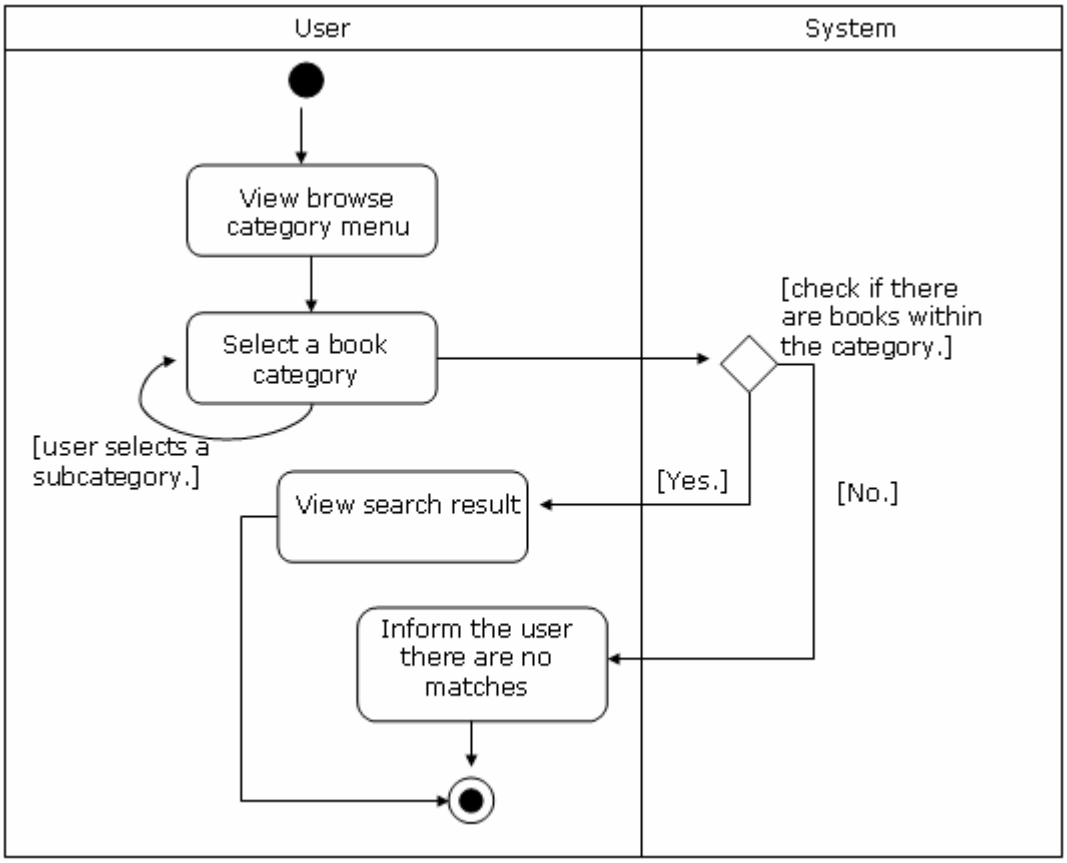


Figure 9.9: Activity Diagram - Browse Category

View Book Details

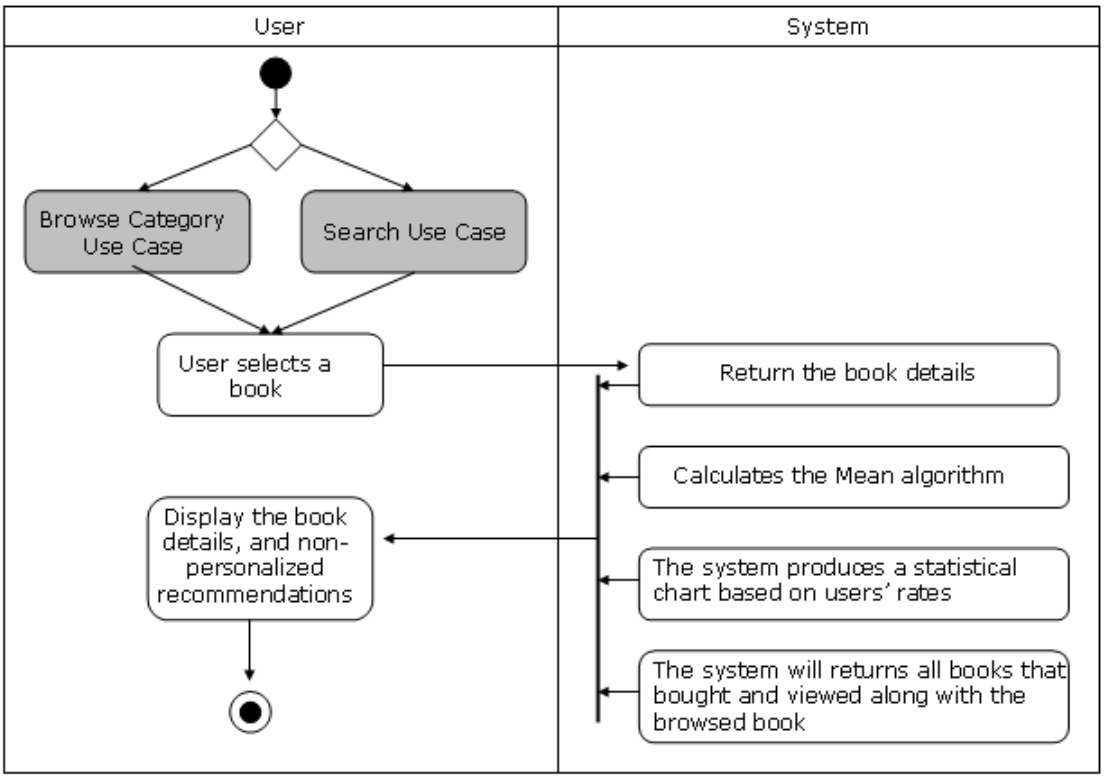


Figure 9.10: Activity Diagram - View Book Details

User Profile:

Rating

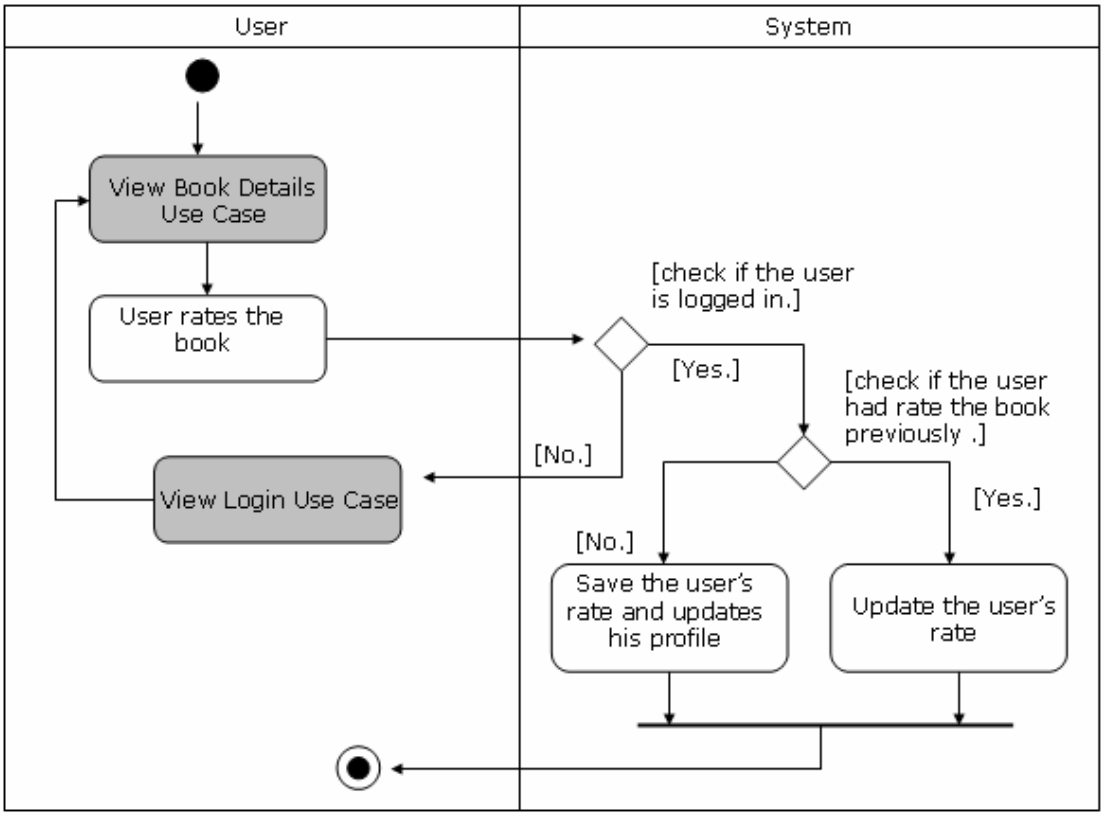


Figure 9.11: Activity Diagram - Rating

Add to Favourite

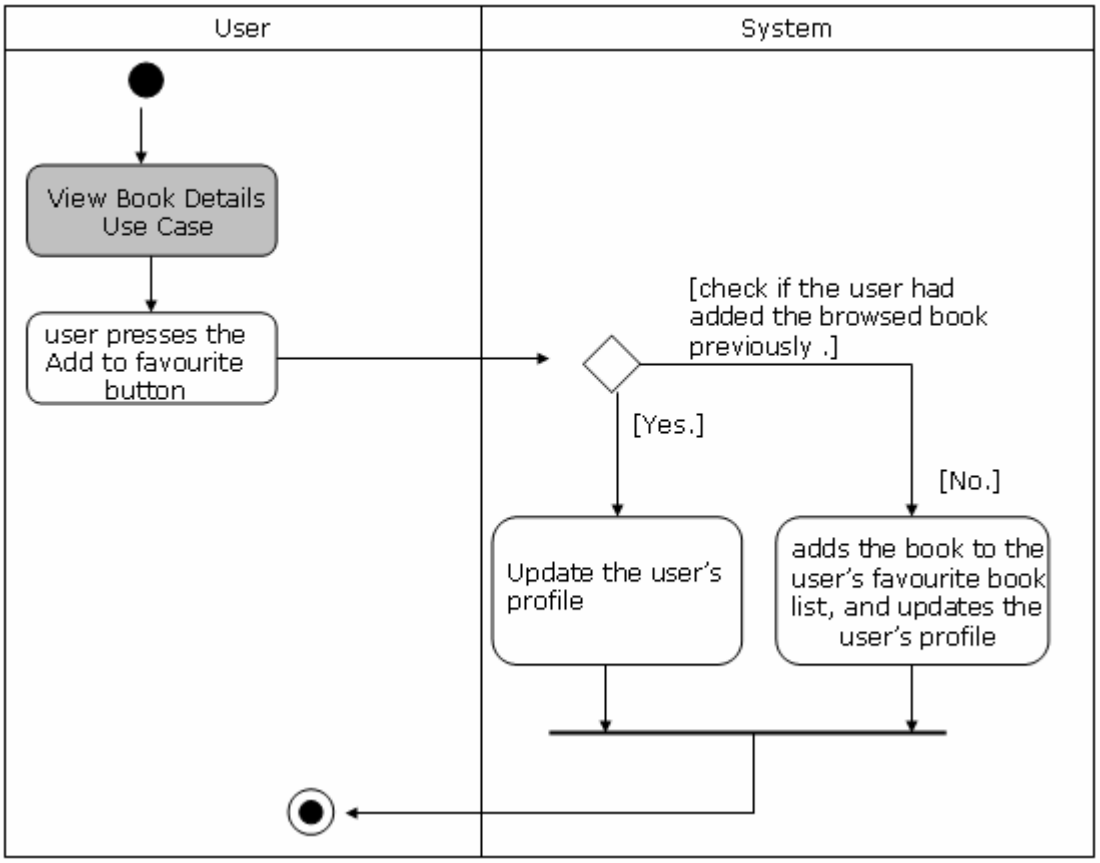


Figure 9.12: Activity Diagram - Add to Favourite

Add to Shopping Cart

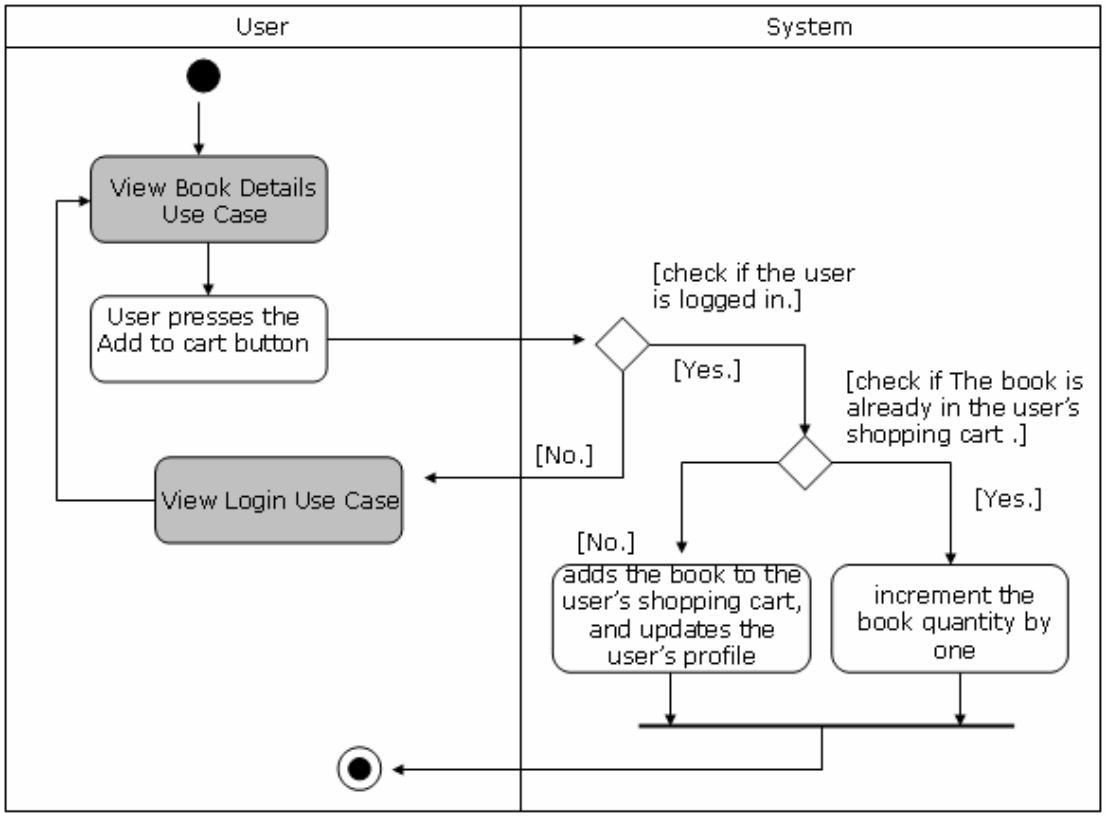


Figure 9.13: Activity Diagram - Add to Shopping Cart

Add to Owned Books List

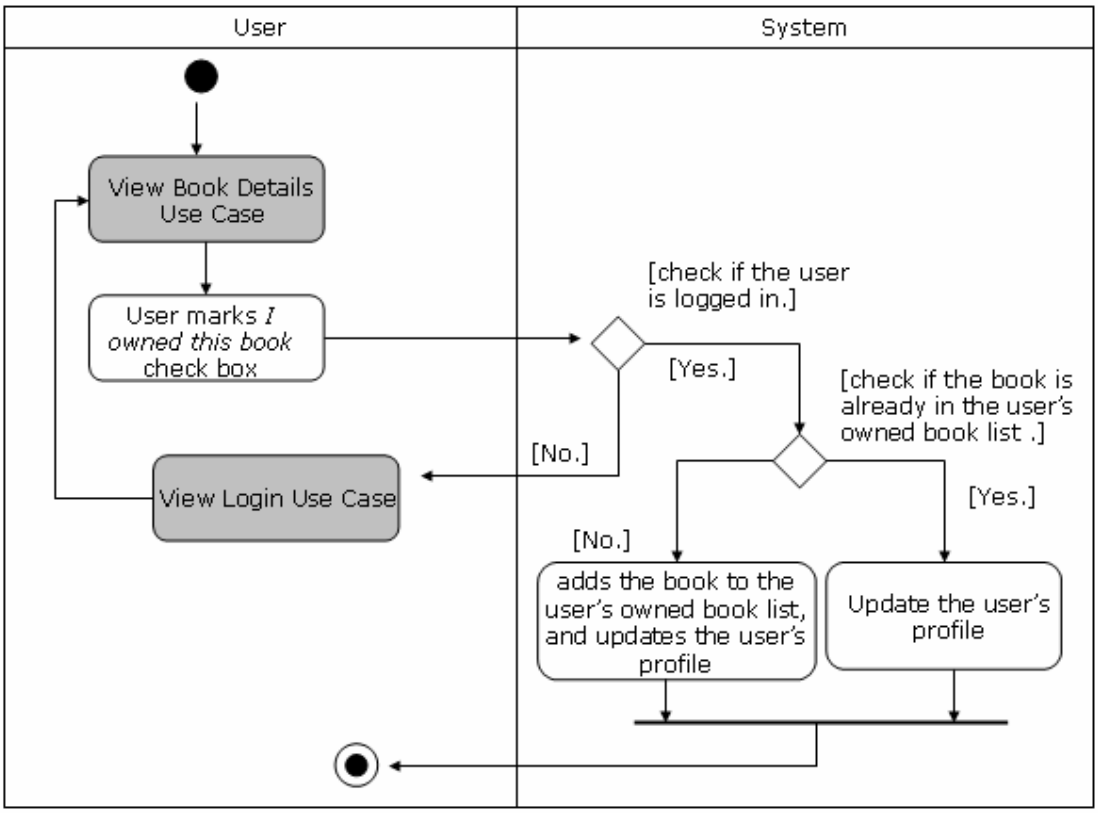


Figure 9.14: Activity Diagram - Add to Owned Books List

Improve Recommendation Area:

View Rated Books

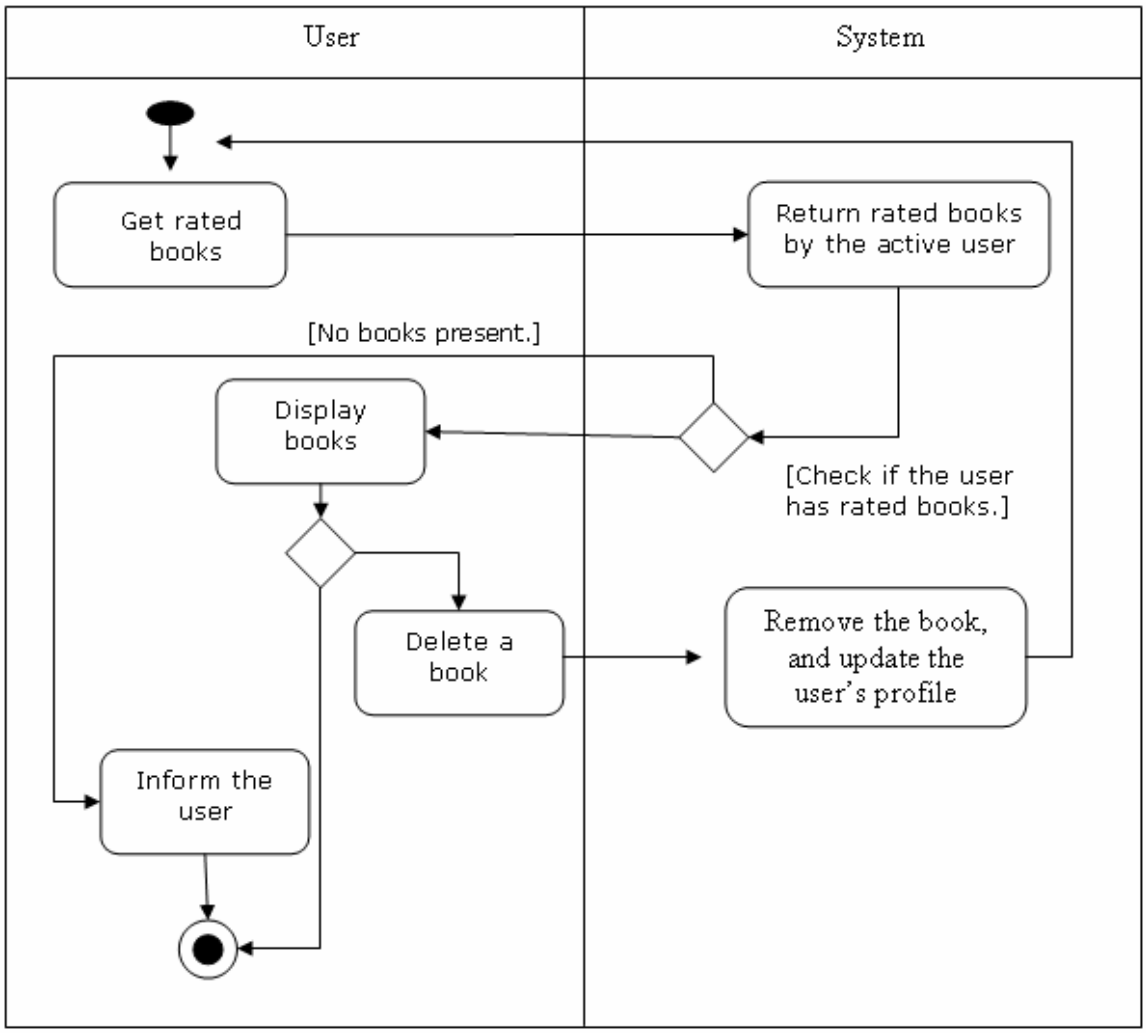


Figure 9.15: Activity Diagram - View Rated Books

View Favourite Books

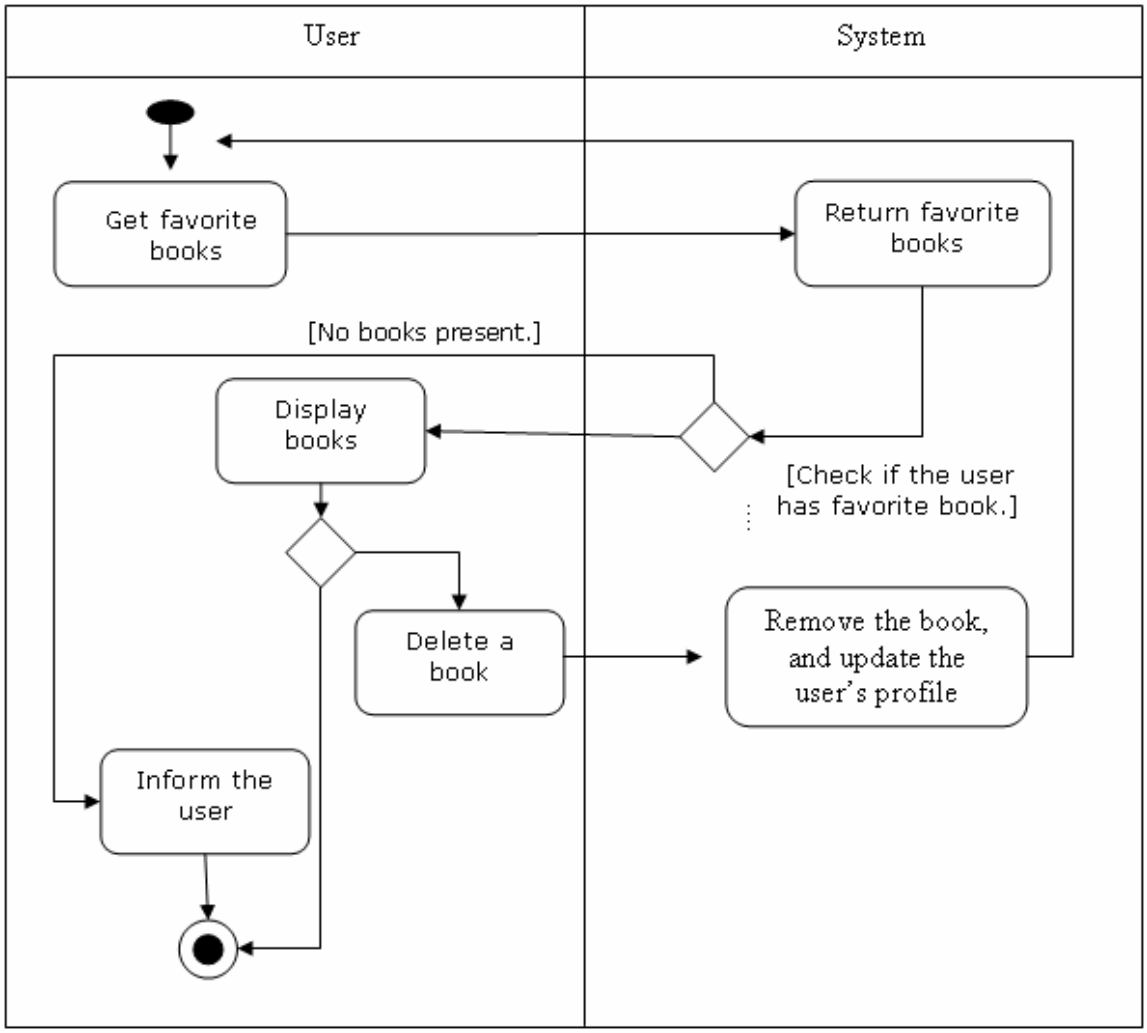


Figure 9.16: Activity Diagram - View Favourite Books

View Owned Books

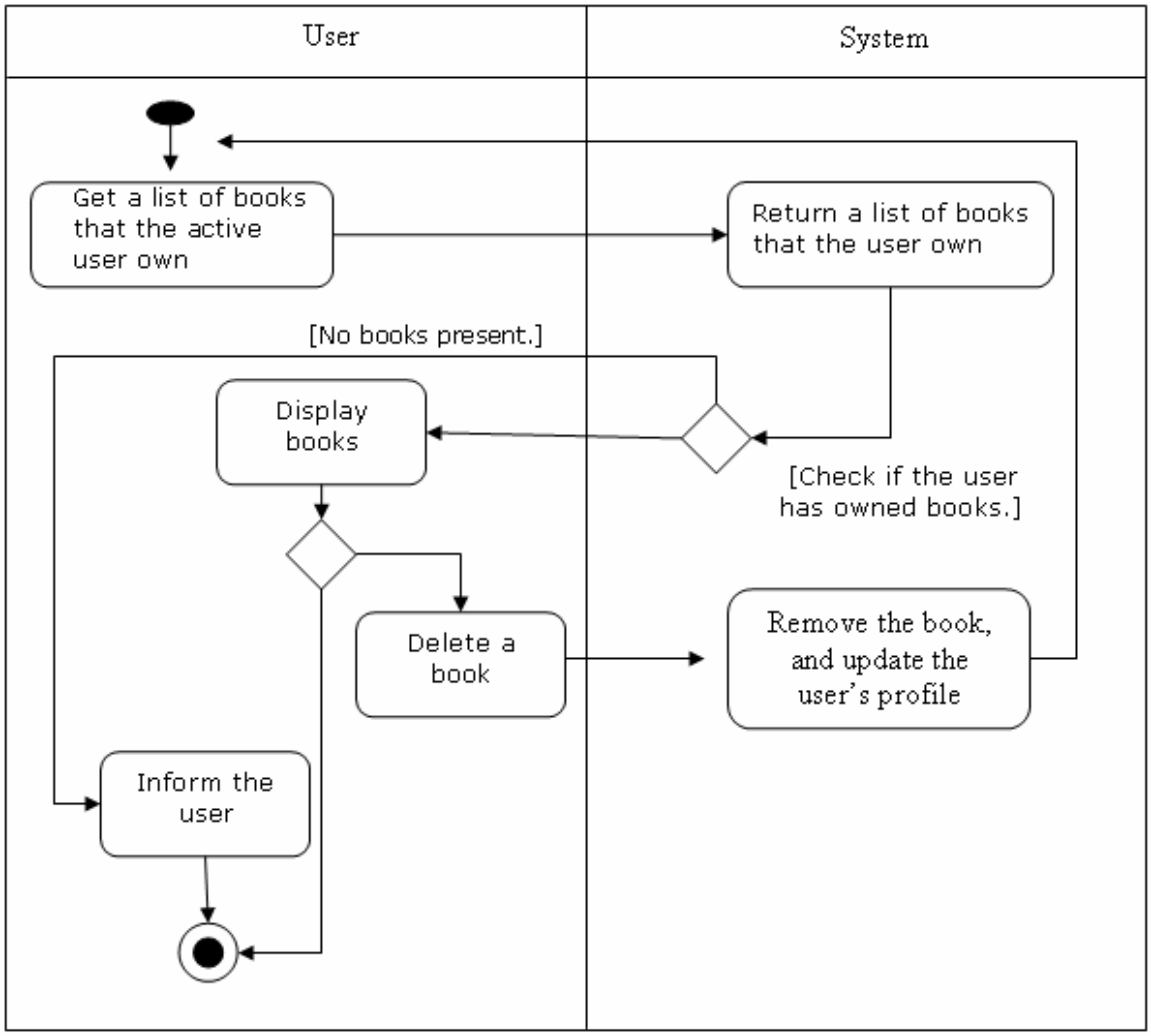


Figure 9.17: Activity Diagram - View Owned Books

View User's Browsing History

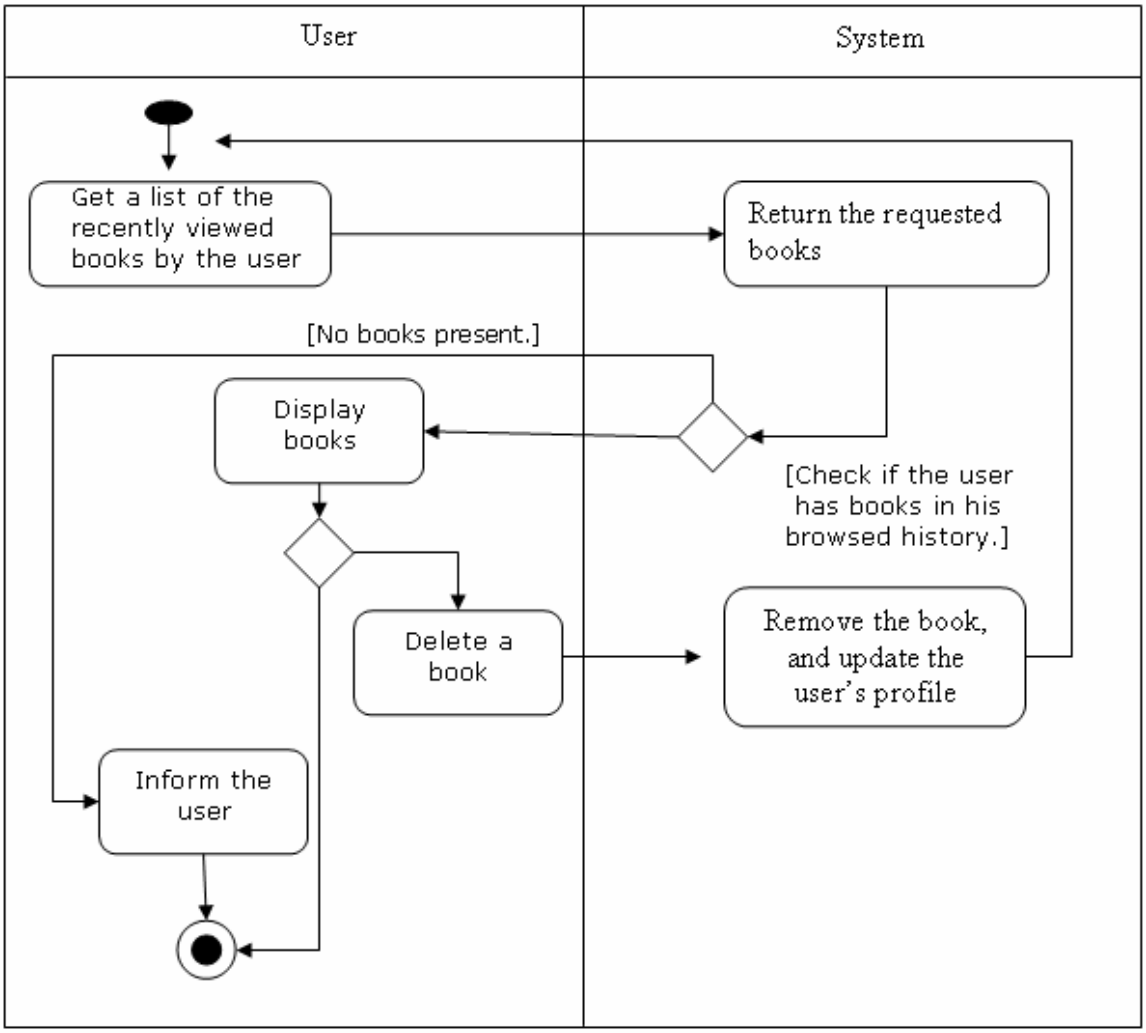


Figure 9.18: Activity Diagram - View User's Browsing History

Recommendation Module:

Content-based Filtering Approach

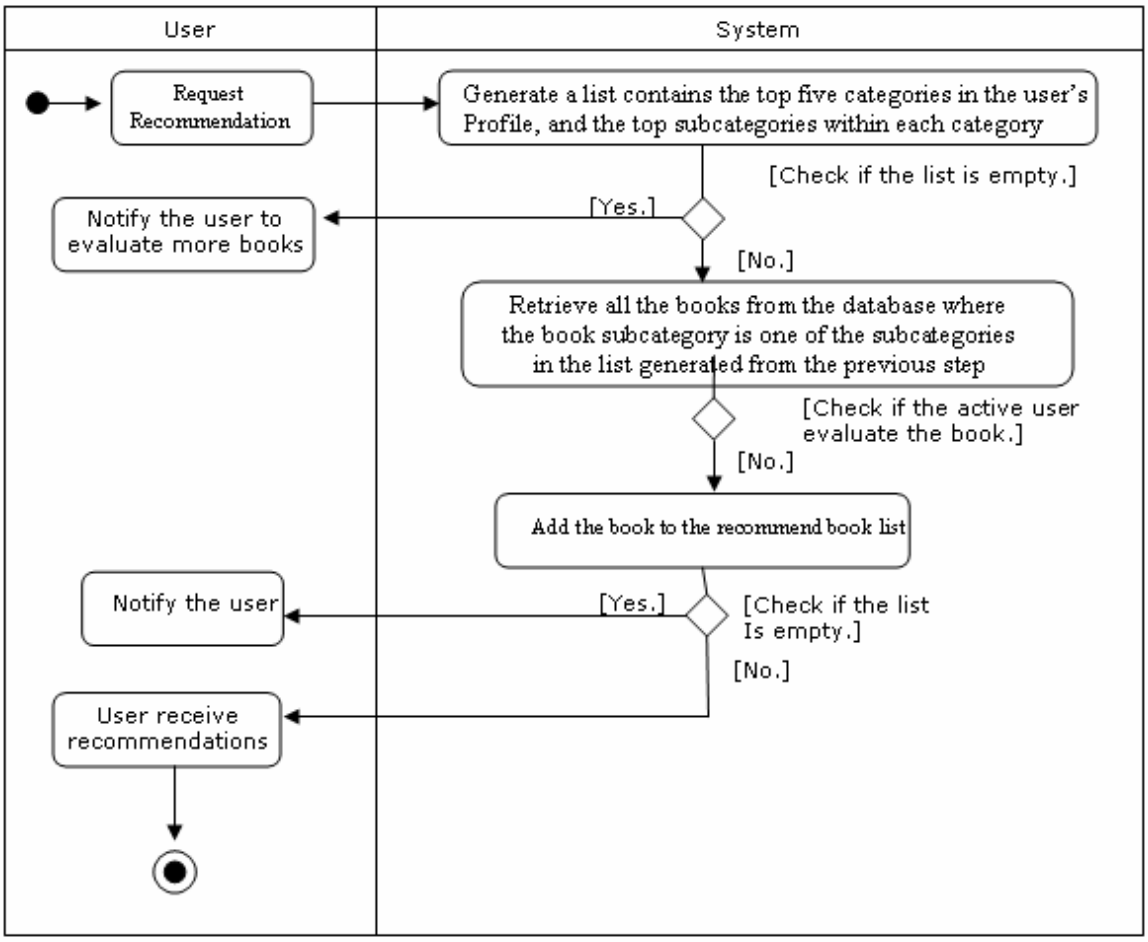


Figure 9.19: Activity Diagram - Content-Based Approach

Collaborative Filtering Approach

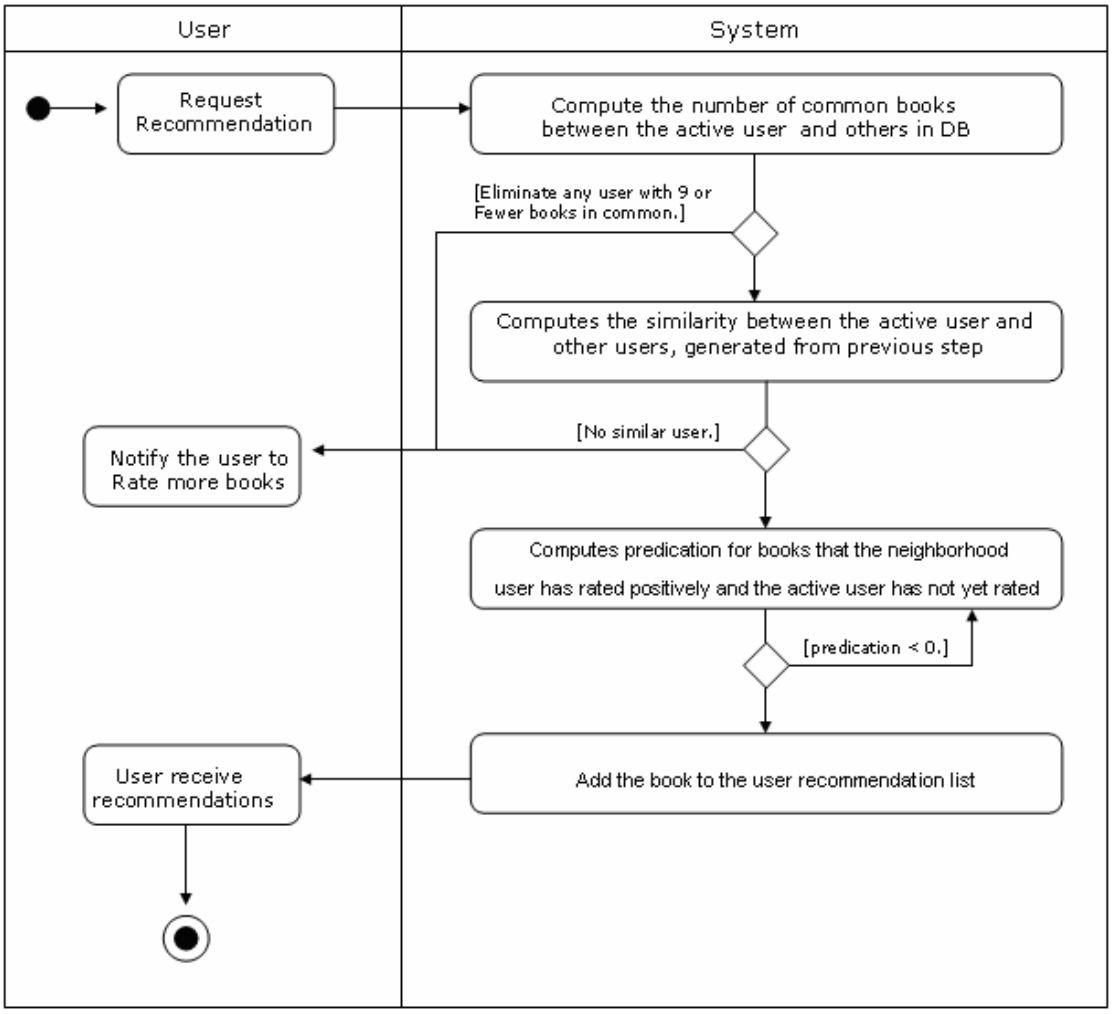


Figure 9.20: Activity Diagram - Collaborative Filtering Approach

Hybrid Filtering Approach

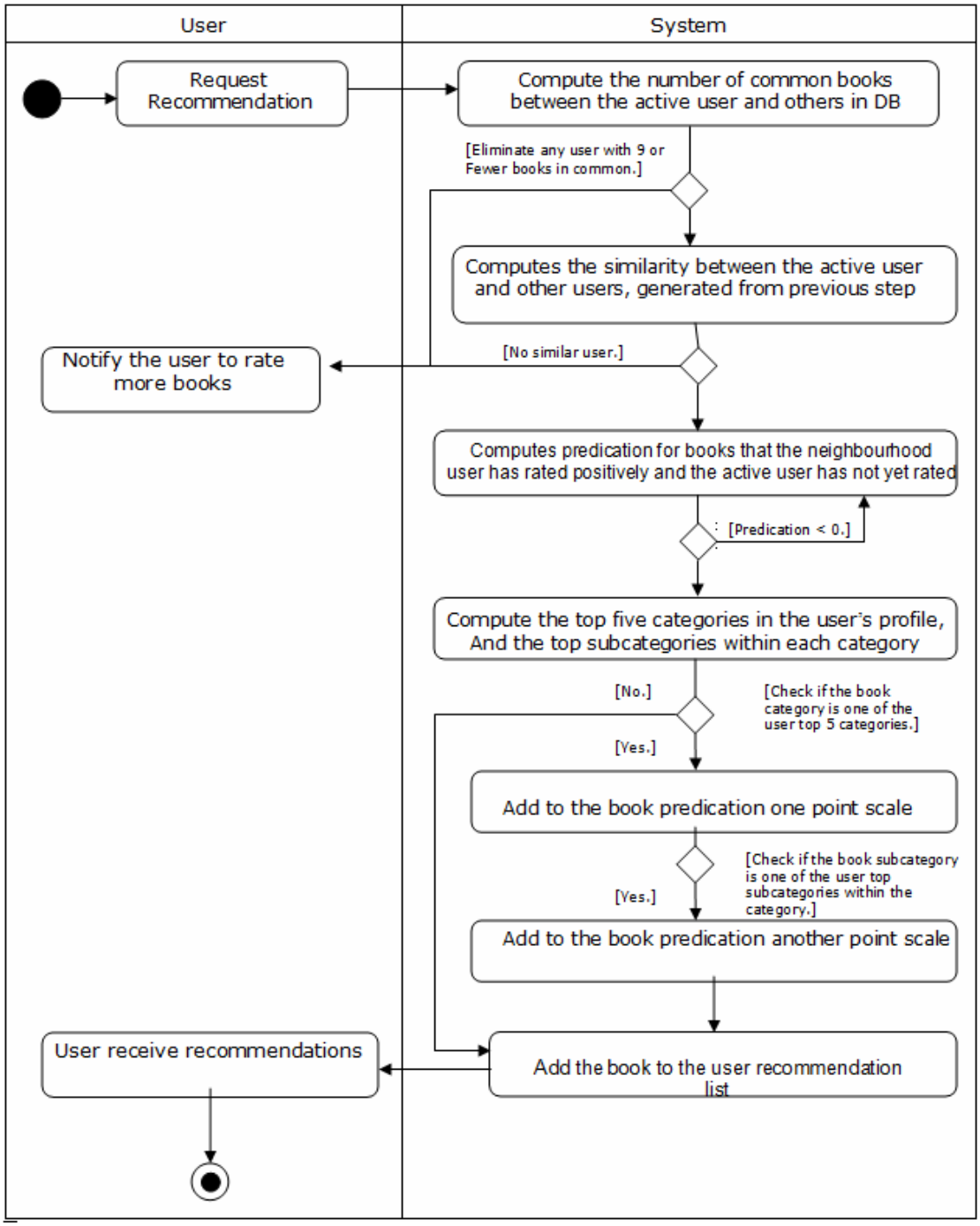


Figure 9.21: Activity Diagram - Hybrid Filtering Approach

9.5 Appendix E: Recommendation Module Class Diagram

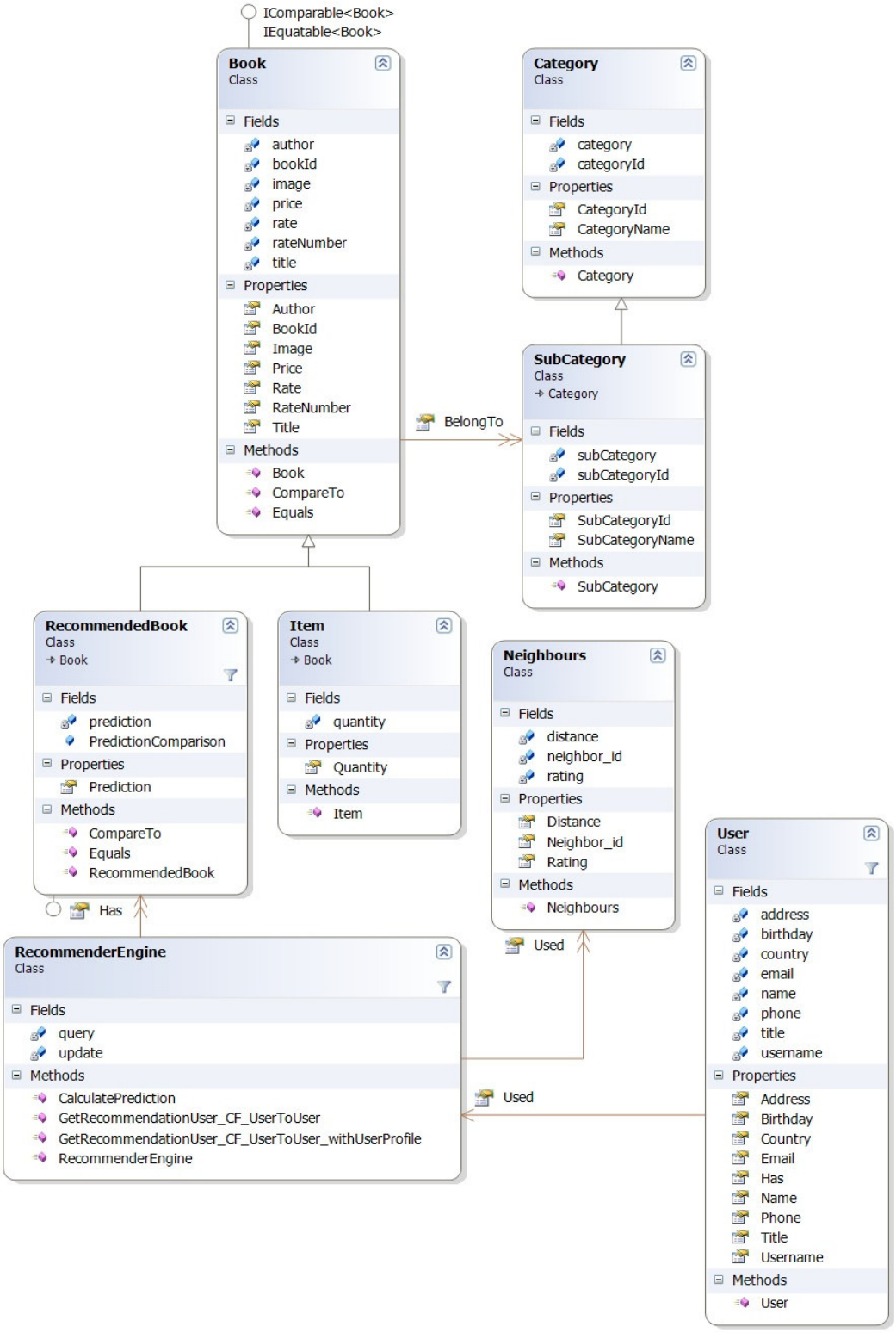


Figure 9.22: Recommendation Module Class Diagram

9.6 Appendix F: Entity-Relationship (ER) Diagram

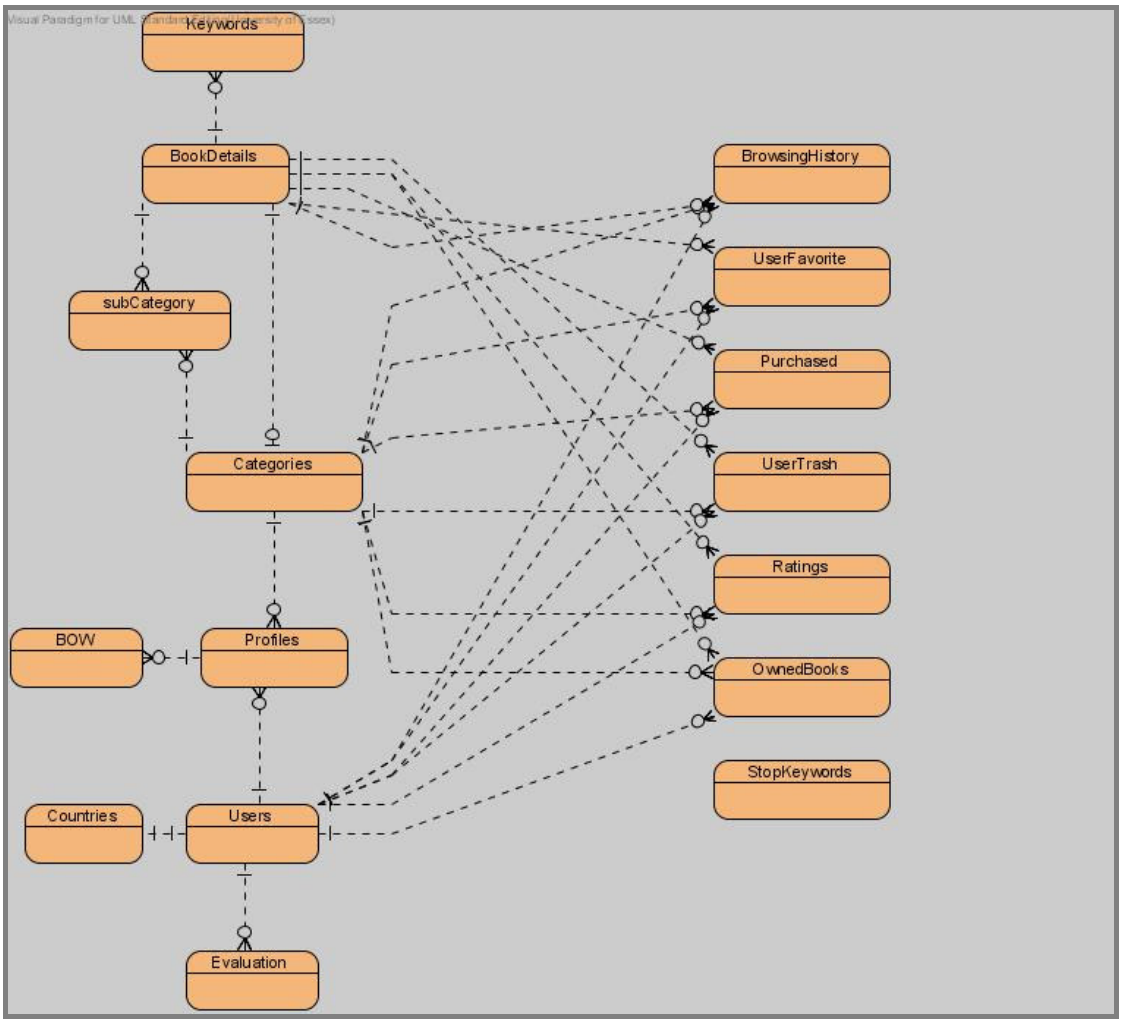


Figure 9.23: ER Diagram

9.7 Appendix G: Relationships within the ER Diagram

Users Table

- ONE **User** can own MANY **OwnedBooks.**
- ONE **User** can have MANY **BrowsingHistory**.
- ONE **User** can have Many **Ratings**.
- ONE **User** can have MANY **UserFavourite**.
- ONE **User** can have MANY **Purchased**.
- ONE **User** can have MANY **Profiles**.
- ONE **User** can have MANY **UserTrash**.
- ONE **User** can have MANY **Evaluations**.
- MANY **Users** belong to ONE **Country**.

BookDetails Table

- MANY BookDetails belong to ONE **Category**.
- ONE **BookDetails** may have MANY **Ratings**.
- ONE **BookDetails** can have MANY **UserFavourite**.
- ONE **BookDetails** can have MANY **OwnedBook**.
- ONE **BookDetails** can have MANY **BrowsingHistory**.
- ONE **BookDetails** can have MANY **Purchased**.
- ONE **BookDetails** can have MANY **UserTrash**.
- ONE **BookDetails** has MANY **Keywords**.

Categories Table

- ONE **Category** contains MANY **BookDetails.**
- ONE **Category** contains MANY **Subcategories.**
- ONE **Category** can have MANY **BrowsingHistory.**
- ONE **Category** can have MANY **Profiles.**
- ONE **Category** can have MANY **OwnedBook.**
- ONE **Category** can have MANY **Ratings.**
- ONE **Category** can have MANY **UserFavourite.**
- ONE **Category** can have MANY **Purchased.**
- ONE **Category** can have MANY **UserTrash.**

SubCategory Table

- MANY **SubCategory** belong to ONE **Categories**.

Ratings Table

- MANY **Ratings** Owend by ONE **Users**.
- MANY **Ratings** belong to ONE **BookDetails**.
- MANY **Ratings** belong to ONE **Categories**.

UserFavourite Table

- MANY **UserFavourite** owned by ONE **Users**.
- MANY **UserFavourite** contain ONE **BookDetails**.
- MANY **UserFavourite** belong to ONE **Categories**.

BrowsingHistory Table
- MANY **BrowsingHistory** owned by ONE **Users**.
- MANY **BrowsingHistory** contain ONE **BookDetails**.
- MANY **BrowsingHistory** belong to ONE **Category**.

Profiles Table
- MANY **Profiles** owned by ONE **User**.
- MANY **Profiles** belong to ONE **Category**.
- ONE **Profile** has MANY **BOW**.

OwnedBook Table
- MANY **OwnedBook** owned by ONE **User**.
- MANY **OwnedBook** belong to ONE **Category**.
- MANY **OwnedBook** contain ONE **BookDetails**.

Purchased Table
- MANY **Purchased** owned by ONE **Users**.
- MANY **Purchased** belong to ONE **Category**.
- MANY **Purchased** contain ONE **BookDetails**.

UserTrash Table
- MANY **UserTrash** owned by ONE **Users**.
- MANY **UserTrash** belong to ONE **Category**.
- MANY **UserTrash** contain ONE **BookDetails**.

Countries Table
- ONE **Country** can have MANY **Users**.

Keywords Table
- Many **Keywords** belong to ONE **BookDetails**.

BOW Table
- Many **BOW** belong to ONE **Profile**.

Evaluation Table
- Many **Evaluations** done by ONE **User**.

9.8 Appendix H: Tables and their Attributes

Table Name: Users	
Attributes	**Data Type**
UserID (PK)	INT AUTO_INCREMENT
userTitle	VARCHAR(10)
name	VARCHAR(50)
Country	VARCHAR(50)
UserAddress	VARCHAR(50)
Phone	VARCHAR(50)
email	VARCHAR(50)
UserName	VARCHAR(50)

Table 9.1: Database Table – Users

Table Name: BookDetails	
Attributes	**Data Type**
bookId (PK)	INT AUTO_INCREMENT
title	VARCHAR(50)
author	VARCHAR(50)
publisher	VARCHAR(50)
year	INT
ISBN	INT
pages	INT
image	VARCHAR(MAX)
imageLarge	VARCHAR(MAX)
category	VARCHAR(50)
language	VARCHAR(50)
price	MONEY
quantity	INT
description	VARCHAR(MAX)
categoryId(FK)	INT
subCategoryId(FK)	INT
subCategoryId2(FK)	INT
subCategoryId3(FK)	INT

Table 9.2: Database Table - BookDetails

Table Name: Categories	
Attributes	**Data Type**
categoryID (PK)	INT AUTO_INCREMENT
category	VARCHAR(50)

Table 9.3: Database Table – Categories

Table Name: SubCategory	
Attributes	**Data Type**
subCategoryId(PK)	INT
subCategory	VARCHAR(50)
categoryId(FK)	INT

Table 9.4: Database Table - SubCategory

Table Name: Ratings	
Attributes	**Data Type**
userID(PK)	INT
bookID(PK)	INT
categoryID(PK)	INT
Rate	SMALLINT
RateTime	DateTime

Table 9.5: Database Table -Ratings

Table Name: UserFavourite	
Attributes	**Data Type**
userID (PK)	INT
bookID (PK)	INT
categoryID(PK)	INT

Table 9.6: Database Table - UserFavourite

Table Name: BrowsingHistory	
Attributes	**Data Type**
userID (PK)	INT
bookID (PK)	INT
categoryId(PK)	INT
visitedDate	DateTime

Table 9.7: Database Table – BrowsingHistory

Table Name: OwnedBook	
Attributes	**Data Type**
userID (PK)	INT
bookID (PK)	INT
categoryId(PK)	INT

Table 9.8: Database Table – OwnedBook

Table Name: Purchased	
Attributes	**Data Type**
userID (PK)	INT
bookID (PK)	INT
categoryId(PK)	INT

Table 9.9: Database Table – Purchased

Table Name: UserTrash	
Attributes	**Data Type**
userID (PK)	INT
bookID (PK)	INT
categoryId(PK)	INT

Table 9.10: Database Table - UserTrash

Table Name: Profiles	
Attributes	**Data Type**
userID (PK)	INT
categoryId (PK)	INT
frequent	INT
BOWID(FK)	INT

Table 9.11: Database Table - Profiles

Table Name: Countries	
Attributes	**Data Type**
Country	VARCHAR(50)
PhoneCode	VARCHAR(10)

Table 9.12: Database Table - Countries

Table Name: StopKeywords	
Attributes	**Data Type**
word (PK)	VARCHAR(50)

Table 9.13: Database Table – StopKeywords

Table Name: Keywords	
Attributes	**Data Type**
BookID (PK)	INT
word (PK)	VARCHAR(50)
frequent	INT

Table 9.14: Database Table – Keywords

Table Name: BOW	
Attributes	**Data Type**
BOWID (PK)	INT
word (PK)	VARCHAR(50)
Jenningsght	INT

Table 9.15: Database Table – BOW

Table Name: Evaluation	
Attributes	**Data Type**
AlgorithmID	INT
userID	INT
trueRate	INT
prediction	INT

Table 9.16: Database Table – Evaluation

9.9 Appendix I: Database Increment Class Diagram

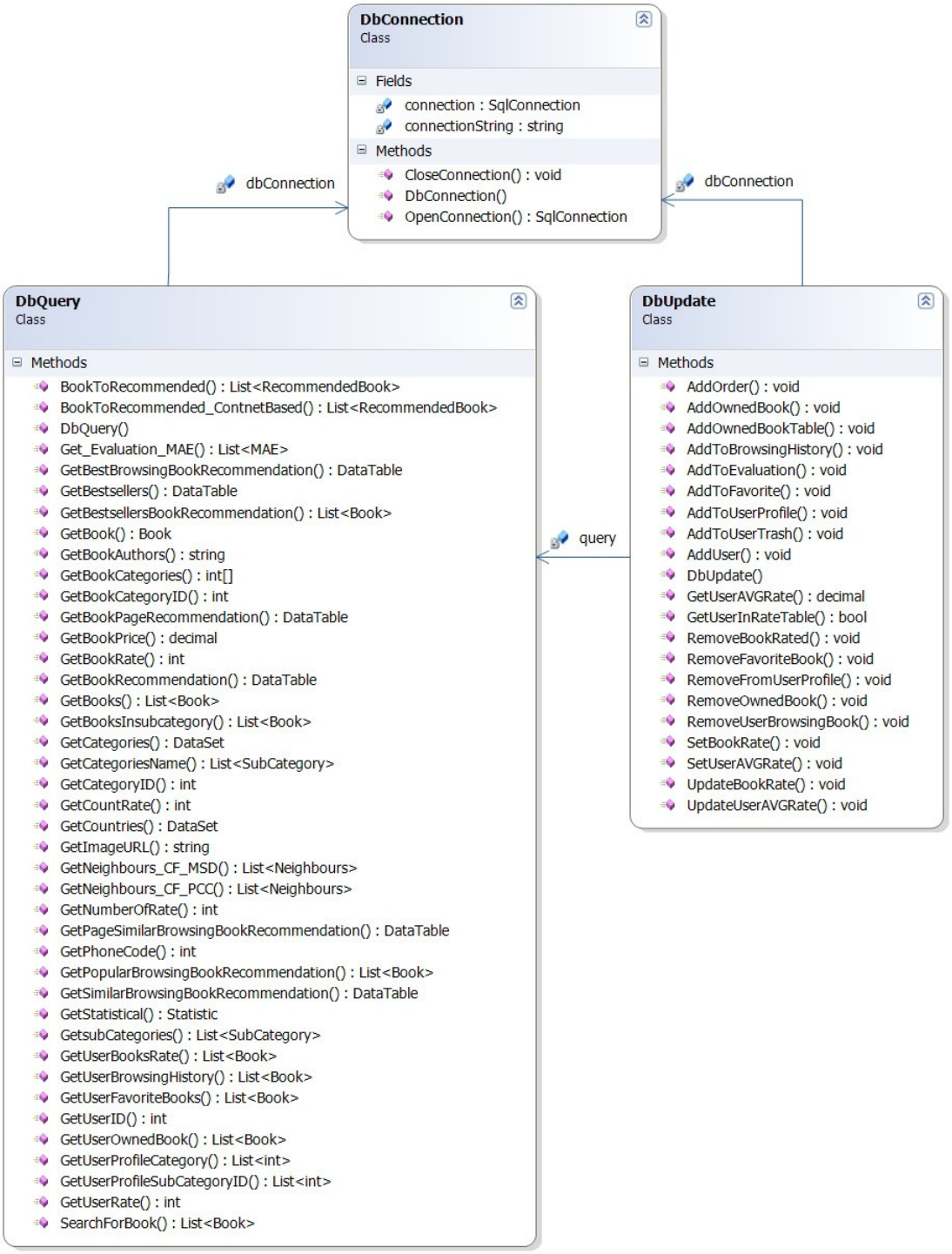


Figure 9.24: Database Class Diagram

9.10 Appendix J: Front End Increment Implementation

This increment structures the main functionality to be used in both the learning module and the recommender module. In this section I will describe the important functions in this increment.

Book Keywords

The book identifier keywords are added to the `Keywords` table to be used for building users profiles and for the search feature. This was done using machine learning techniques. Once the website administrator adds a book, the `CountKeywords` method is invoked. This method processes the book's title, author and description by removing the stop words using the words from the `StopKeywords` table. After that, the method gives a weight for each of a book's words, which represents the number of times the word appears in the book's title or description. This is done with Dictionary generic class, which provides a mapping from a set of keys to a set of values. The book's words form the set of keys, while the weight values represent the set of values. Finally, these words, along with their weights values, are inserted into the `Keywords` table associated with the book ID as an identifier.

Categories Navigation

One of the important keys of the design is that each book belongs to more than one subcategory within the book category. This design is similar to the tree structure, where the book category is a root node in the hierarchy and the subcategories of the book form sub-nodes. For example, a book about PHP belongs to the Computers category and to Programming, Database and Website Design subcategories. **Figure 9.25** shows an example of the navigation of the categories.

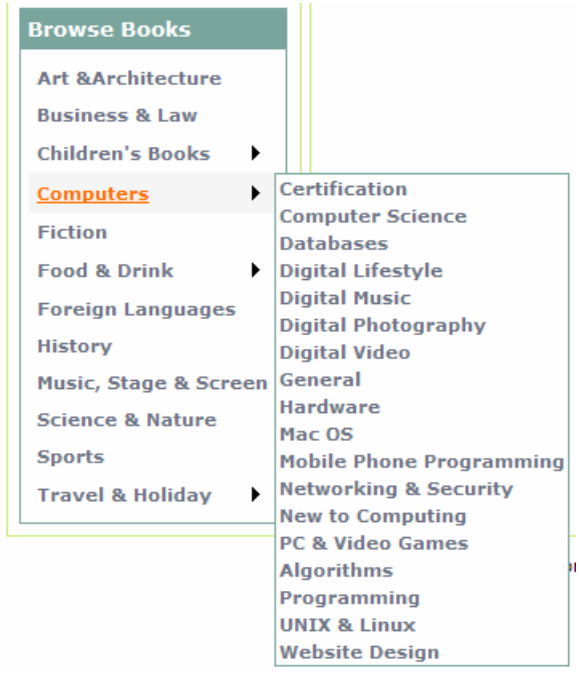


Figure 9.25: Category Browsing

The navigation control shows the categories as a link to enable the user to view all the books within the category. Also, each category has a child menu which contains all the subcategories within the category. This is done by calling `GetsubCategories` method to retrieve the subcategories of the current category. Each subcategory is represented as a link which connects the user to all the books belonging to the subcategory. The categories are represented in a way that helps the user understand the hierarchical nature of the categories.

When the user visits a category or a subcategory, the categories navigation will change to display all the subcategories within the visited category, as shown in **Figure 9.26**.

Figure 9.26: Subcategories Navigation

This is done with the `LoadSubCategor` method, which uses the `categoryID` query string to retrieve all the subcategories within the category, then draws the external borders for the navigation menu, and then draws the internal borders. After that, each subcategory is represented in the navigation menu as a hyperlink to connect the user to all the books belonging to the subcategory.

Categories Browsing

The system provides categories browsing, which permits users to easily find books without having to use a search feature. When the user clicks a particular category on the navigation menu, the Browse page loads. This calls the `GetBooks` method in the `DbQuery` class, and the `categoryID` query string is passed as a parameter. After that, a `DataTable` is created containing the results set to be bound to the `GridView` control which is the standard way of displaying a list of records on a web page using ASP.NET. For each record in the `DataTable`, a single template field is used within the `GridView`, rather than letting the control display the fields in the default view. In addition, HTML is used inside the `ItemTemplate` to define the appearance of the book object, and `'<%# Eval("FieldName") %>'` syntax is used to displayed the fields in the `ItemTemplate` from the results set. The book title and image are displayed with a hyperlink to the `bookDetails.aspx` page. Finally, the `bookID` is inserted into the hyperlink URL using the `'<%# "bookDetails.aspx?bookID="+ Eval("bookID") %>'` syntax.

Search

The application provides simple and advanced search for the users to search for books. The simple search requires the user to type in a simple keyword which aims for a quick search and a ranked list of results. In contrast, the advanced search is more complex than the simple search. It provides many options for the users, and has the advantage of getting a small result set back.

The simple search consists of a text box and a button, located in the master page, to be accessed by any page on the website. It is important to mention the design considerations for the simple search feature. The most widely used solution in commercial websites for searching purposes is the use of `LIKE` in the `WHERE` clause of the `SELECT` statement. This solution has improved over time, and many weaknesses have been corrected. For example, the search results are returned in random order. Also, this solution is not accurate as the query string returns all the records that have the keyword somewhere in their fields. For example, if the user is looking for *.NET* books and he enters the word 'net' for the keyword field, the following query string returns all the books that have the word 'net' in their title:

```
SELECT * FROM BOOK WHERE Title LIKE '%Keyword%' OR Description LIKE '%Keyword%'
```

For the reasons mentioned, the simple search feature was implemented by the use of the `Keyword` table.

When the user clicks on the search button, the search terms from the text box are appended to the query string with the `keyword` parameter, and then the request is forwarded to the `Results.aspx` page. The keyword parameter is encoded using `Server.UrlEncode` to avoid problems with non-alphanumeric characters. The `Page_Load()` first checks for keyword parameters in the query string. If the keyword is found, the `String.Trim().Split(splitAt)` is called; this divides the keyword paramether into an array of individual strings using an array of different characters to divide the keyword parameter. After that, each keyword is passed as a parameter to the `SimpleSearch` method, which is called the `Proc_SimpleSearch` stored procedure. This stored procedure first checks for the keyword type using the `IsNumeric` function. If the keyword is numeric,

which means the user is searching for a book using the book's ISBN number, the query will select the book from the `BookDetails` table that has the same ISBN number. Otherwise, the query selects all the book IDs from the Keywords table that match the keyword. In addition, the stored procedure returns, along with every book ID, a frequency value representing the number of times the keyword appears in the book's title or description. The selected book IDs are used to select books details from the `BookDetails` table. Finally, the results are returned and ordered based on the frequency value associated with each book in the result.

In contrast, the advanced search does not provide a ranking for the search results, but it has the advantages of providing a small results set. The advanced search form allocated in the `AdvanceSearch.aspx` page consists of four text boxes which take the following parameters: search term, any word appearing in the book's title, the name of the book's author and the book's ISBN. Also, this form consists of one drop-down list representing the book's categories. The user has the option to fill in one or more of the four text boxes and/or select a category to perform a search.

Once the user clicks the search button, an event handler is invoked which checks the text in all four text boxes and the selected value in the drop down list. If the user does not fill any text or select any category, the system will display an error message to inform the user. Otherwise, the keywords from the text boxes and the selected value from the Dropdown list control are appended to the query string with the `keyword, title, author, ISBN` and `categoryID` parameters; then the request is forwarded to the `AdvancedSearchResult` page. These parameters are encoded using `Server.UrlEncode` to avoid any problem with non-alphanumeric characters. It was decided to put the search parameters in the URL to enable users to bookmark the search results page. In the `AdvancedSearchResult.aspx` page, the `Page_Load()` first checks for the five parameters in the query string> If the parameters are found, the values from the query string are decoded using `Server.UrlDecode` which convert the query string parameter from the ASCII equivalent of non-alphanumeric characters back to their original character form. After that, the `SearchForBook` method in the `DbQuery` class is accessed; this call the `Proc_SearchForBook` stored procedure and passed for it the five parameters. This procedure first checks each parameter. If at least one parameter contains a value, a select statement is created to select all the book details from the `BookDetails` table, and then for

each parameter a `WHERE` clause is created dynamically to obtain a condition for the book attribute that matches the parameter type. For example, for the author parameter, the created `WHERE` clause returns all books that the author has written. Finally, the created `WHERE` clauses are added to the select statement to be executed. The results are returned and bind to the `GridView` control to be displayed.

Book Detailed

Details of book are displayed to the user in the `bookDetails.aspx` page; **Figure 9.27** represents the page layout. The `bookID` number is used in the URL to enable users to bookmark book pages. The `BookID` number for the book from the query string is passed as a parameter to the `StProc_GetBook` stored procedure. This will return a `DataTable` containing all the information about the book expected for its average rate and the number of users who rate the book, which is obtained by calling the `GetBookRate` and `GetNumberOfRate` methods. In addition, the authors of the book are retrieved by calling the `GetBookAuthors` method in the `DBQuery` class, and then displayed as hyperlinks. (These hyperlinks are described in the next section.) The data from this `DataTable` are assigned to the various controls in the `DetailsView,` which displays details of the book. Finally, the Add to Basket button is displayed by default, while the Add to Favourite button is only displayed if the user is logged in.

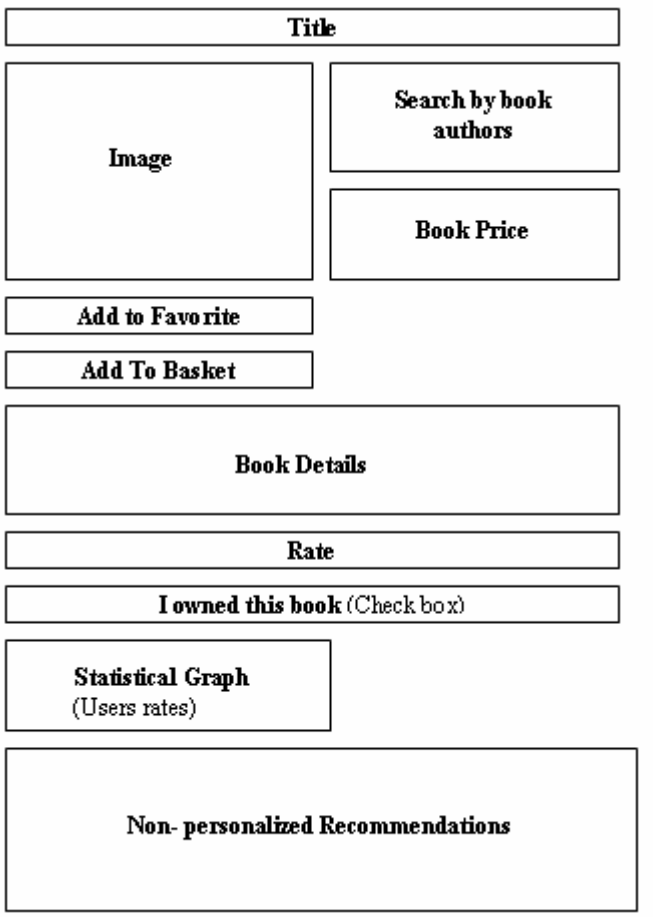


Figure 9.27: Page Layout for Book Details

Search by Author Name

Search by author name enables users to find other books written by the author of the browsed book. For example, when a book is displayed in the `bookDetails.aspx`, the names of the book's authors will be shown as hyperlinks. Each author hyperlink contains a link to the `Results.aspx` page, with an author name parameter in the query string. Once one of these hyperlinks is selected, the user will be taken to the `Results.aspx` page, which checks for the keyword parameter in the query string. If the keyword is found, `SearchForBook` method is called, which returns a list of books the selected author has written.

User's Preferences Graph

The users can view a graph that represents their interests in a particular category. This is done in the user control panel when the user clicks the statistical hyperlink. A form appears which request from the user to select a category to show statistical data about the user's preferences in the selected category. When the user selects a particular category and then clicks the Show

button, the `ShowGraph` method is invoked which accesses the `GetUserStatistical` method in the statistical class. This method works in three steps:

1. The `GetsubCategories(categoryID)` method is accessed to retrieve all the subcategories within the selected category.
2. For loop travel, the subcategories, and for each subcategory, the `GetStatistical` method is accessed to calculate three values: the number of books the user rated, the number added as a favourite, and the number of books the user owned.
3. Finally, the system checks if there are any books that the user evaluated in the selected category; if so, a message will display to inform the user. Otherwise, the system will draw the graph using Dundas Chart Template and will group the column chart by the subcategory, across the number of books, as the graph series data. **Figure 9.28** shows an example of a graph in action.

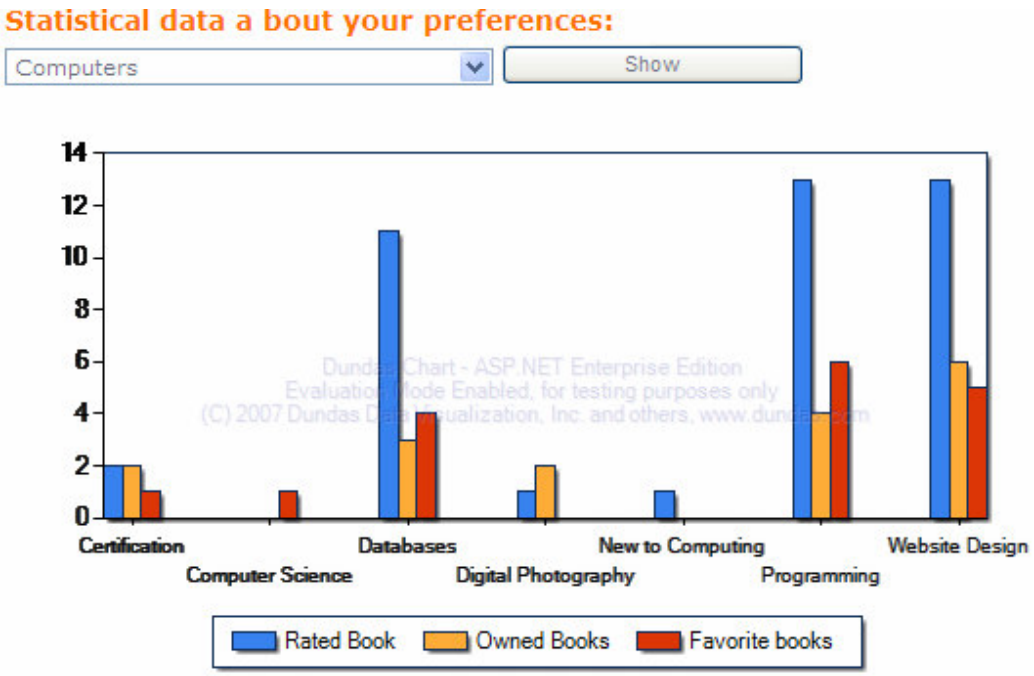


Figure 9.28: User's Preferences Graph

9.11 Appendix K: Unit Testing

Class : Book			
Method	**Input**	**Output**	**Expected Result**
CompareTo	Book object	Comparison of book objects	√
Equals	Book object	Equality of book objects	√
BookId	-	Return bookID	√

Table 9.17: Unit Testing - Book Class

Class : User			
Method	**Input**	**Output**	**Expected Result**
CompareTo	User object	Comparison of user objects	√
Equals	User object	Equality of user objects	√

Table 9.18: Unit Testing - User Class

Class : Basket			
Method	**Input**	**Output**	**Expected Result**
AddItem	Item object	Add an item to the list of items in the user's basket and update the item's quantity.	√
GetAllItems	-	Return all items in the user's basket	√
RemoveItem	ItemID: int	Remove the provided item form the user's basket and update the total price	√
GetQuantity	ItemID:int	Return the quantity of the provided item in the user's basket	√
GetTotalQuantity	-	Return the total quantity of items in the user's basket	√
UpdateQuantity	ItemID:int, quantity:int	Update the quantity of the provided item in the user's basket.	√
Empty()	-	Clear user's basket	√

Table 9.19: Unit Testing - Basket Class

Class : DBQuery			
Method	**Input**	**Output**	**Expected Result**
GetCountries	-	Return a list of courtiers' name from the Countries table	√
GetPhoneCode	Country name	Get the provided courtier's phone code from the Countries table	√
GetUserID	User name	Return the user ID	√
GetBooks	Category ID	Return all books belong to the category ID provided	√
GetBook	Book ID	Return the book details for the book ID provided	√
GetBooksInsubcategory	Sub category ID	Return books belong to the sub category provided	√
GetImageURL	Book ID	Return the book's image URL for the book ID provided	√
GetBookPrice	Book ID	Return the book's price for the book ID provided	√
GetBookCategoryID	Book ID	Return the book's category ID	√
GetBookRate	Book ID	Return the average rating for the provided book	√
GetStopKeywords	-	Return a list of stop words from the database	√
GetCountRate	Book ID, rate	Return the number of users who provided this rate for the provided book	√
GetUserBooksRate	User ID	Return a list of books rated by the user	√
GetUserFavouriteBooks	User ID	Return a list of user's favourite books from the database	√

Table 9.20: Unit Testing - DBQuery Class (1)

Class : DBQuery			
Method	**Input**	**Output**	**Expected Result**
GetUserBrowsingHistory	User ID	Return all books in the user's browsing history from the database	√
GetNumberOfRate	Book ID	Return the number of users who provided rate for the book	√
GetUserOwnedBook	User ID	Return all books that the user owned from the database	√
GetCategories	-	Return all the categories from the database	√
GetsubCategories	Category ID	Return a list of subcategories belong to the category provided	√
GetUserRate	User ID and Book ID	Return the provided user's rate for the provided book	√
GetBookKeywords	Book ID	Return the provided book's keywords from the database	√
GetBOWID	User ID, category ID ,sub category ID	Return the BOW ID within the provided parameters.	√
GetStatistical	User ID, category ID, sub category ID	Return a statistical data for the user within the categories provided	√
GetBookRecommendation	Book ID, category ID	Return a list of books form the database where the book bought with the provided book	√
GetSimilarBrowsingBookRecommendation	Book ID, category ID	Return a list of books form from the database where the book browsed with the provided book.	√

Table 9.21: Unit Testing - DBQuery Class (2)

Class : DBQuery Continued			
Method	**Input**	**Output**	**Expected Result**
GetBestBrowsingBook Recommendation	-	Return The best browsed books based on the users' browsing history	√
GetBestsellers	-	Return The best Bestsellers books	√
SearchForBook	Keyword, title, author, ISBN, category ID	Return a collection of books match the provided parameters	√
SimpleSearch	keyword	Return collection of books match the keyword provided	√
GetBookAuthors	Book ID	Return a book's authors within the book ID provided	√
GetNeighbours_CF_MSD	User ID, threshold, the minimum common books	Calculate the neighbours for the provided user using MSD algorithm with a minimum books in common and higher than a threshold	√
GetNeighbours_CF_PCC	User ID, threshold, the minimum common books	Calculate the neighbours for the provided user using Pearson correlation with a minimum books in common and higher than a threshold	√

Table 9.22: Unit Testing - DBQuery Class (3)

Class : DBQuery Continued			
Method	**Input**	**Output**	**Expected Result**
GetUserProfileCategory	User ID	Construct a user profile based on the frequency of categories	√
GetUserProfileSubCategoryID	User ID , category ID	Construct a user profile based on the frequency of sub categories within the provided category	√
BookToRecommended	User ID , neighbor ID	Return a list of books that the neighbour user his rated highly and the provided user has not evaluated	√
BookToRecommended_ContnetBased	User ID, category ID , subcategory ID	Return a collection of books that the provided user did not evaluate within the provided category and sub category	√
Get_Evaluation_MAE	-	Calculates the Mean Absolute Error for each algorithm.	√

Table 9.23: Unit Testing - DBQuery Class (4)

Class : DBUpdate			
Method	**Input**	**Output**	**Expected Result**
AddUser	User object	Insert a user into the database	√
AddBook	Book object	Insert a book into the database	√
AddToFavourite	Book ID, user ID and category ID	Insert a book ID and category into the provided user's favourite in the database.	√
AddToBrowsingHistory	Book ID, user ID ,category ID	Insert a book ID and category into the provided user's browsing history in the database.	√
RemoveFavouriteBook	User ID, Book ID	Delete the provided book form the user's favourite in the database	√
RemoveUserBrowsingBook	User ID, Book ID	Delete the provided book form the user's browsing history in the database	√

Table 9.24: Unit Testing - DBUpdate Class (1)

Class : DBUpdate			
Method	**Input**	**Output**	**Expected Result**
RemoveBookRated	User ID, Book ID	Delete the user's rating for the provided book	√
RemoveOwnedBook	User ID, Book ID	Delete the provided book from the user's owned book in the database	√
SetBookRate	User ID, book ID, category ID, rating	Insert a book rating into the database	√
UpdateBookRate	User ID, book ID, category ID, rating	Update the user's rating for the provided book.	√
GetUserAVGRate	User ID	Return the average rate for the provided user	√
AddOrder	User ID, list of items object	Insert the provided user's purchased into the database.	√

Table 9.25: Unit Testing - DBUpdate Class (2)

Class : DBUpdate			
Method	**Input**	**Output**	**Expected Result**
AddOwnedBookTable	User ID, list of items object	Insert the provided user's purchased into his owned books list in the database.	√
AddOwnedBook	User ID, item object	Insert a book into the provided user's owned books list in the database.	√
AddToUserProfile	User ID, category ID, sub category ID	Insert into the provided user's profile the provided categories. If the categories exist in the user's profile increment the associate frequent value by one.	√
RemoveFromUserProfile	User ID, category ID, sub category ID	Decrement the frequent value in the user's profile within the provided categories.	√
AddToUserTrash	User ID, Book ID, category ID	Insert into the user's trash in the database the provided book.	√
AddToEvaluation	Evaluation object	Insert the provided evaluation details into the database.	√

Table 9.26: Unit Testing - DBUpdate Class (3)

Class : DBUpdate			
Method	**Input**	**Output**	**Expected Result**
AddKeywords	Book ID, Dictionary of keywords	Insert into the database the provided book's keywords along with a frequent value for each keyword	√
AddToBOW	BOW ID, Dictionary of keywords	Insert into the user's profile the provided book's keywords along with a Jenningsght value for each keyword	√
UpdateBOW	BOW ID, Dictionary of keywords	Update the BOW in the user's profile.	√

Table 9.27: Unit Testing - DBUpdate Class (4)

Class : Neighbours			
Method	**Input**	**Output**	**Expected Result**
CompareTo	Neighbours object	Comparison of Neighbours objects	√
Equals	Neighbours object	Equality of Neighbours objects	√

Table 9.28: Unit Testing - Neighbours Class

Class : Today			
Method	**Input**	**Output**	**Expected Result**
createDate	-	Return the current date in the form day month Name year	√
getMonth	Month	Return the name of the corresponding month	√

Table 9.29: Unit Testing -Today Class

Class : RecommenderEngine			
Method	**Input**	**Output**	**Expected Result**
CalculatePrediction	Book ID, collection of neighbours users	Calculate the prediction of the provided book taking into account the neighbour rating.	√
GetRecommendationUser_CF_ UserToUser_withUserProfile	User ID, collection of neighbours users	Run the User-User Nearest Neighbour algorithm with user profile.	√
GetRecommendationUser_CF_ UserToUser	User ID, collection of neighbours users	Run the User-User Nearest Neighbour algorithm.	√

Table 9.30: Unit Testing - RecommenderEngine Class

Class : Taxi			
Method	**Input**	**Output**	**Expected Result**
setSessionStatus	sessionStatus : Boolean	Set the user's status.	√
getSessionStatus	-	Return the user's status.	√
setUserType	User type	Set the user's type (user or admin).	√
getUserType	-	Return the user's type.	√
setUserID	User ID	Set the user's ID	√
getUserID	-	Return the user's ID	√
setUsername	User name	Set the user's user name.	√
getUsername	-	Return the user's name.	√

Table 9.31: Unit Testing - Taxi Class

Class : AlgorithmEvaluation			
Method	**Input**	**Output**	**Expected Result**
AddToEvaluation	Evaluation object	Insert the provided evaluation details into the database.	√
Get_List_UserRate	Algorithm ID	Returns a list of users' rating along with the rated book ID and the user ID from the evaluation table within the provided algorithm ID	√
Get_Evaluation_MAE	-	Return the performance results of the system algorithms evaluation	√
Experiments_Parameters	Algorithm ID, number of common books, threshold	Return the performance results of the provided algorithm evaluation within the provided parameters	√
Get_List_User	Min vote, max vote	Returns a list of users' rate along with the rated book ID and the user ID from the Ratings table	√
Experiments_NoVotes	Min vote, max vote	Returns the evaluation results of the sensitivity of the number of votes by the active user in collaborative filtering	√
`Experiments_NoVotesHybrid`	Min vote, max vote	Returns the evaluation results of the sensitivity of the number of votes by the active user in hybrid filtering	√

Table 9.32: Unit Testing - AlgorithmEvaluation Class

9.12 Appendix L : Integration Testing

Integrate Database increment with Front End increment			
System operation	**Input**	**Output**	**Expected Result**
Login (valid)	Valid login details	Log to the system	√
Login (invalid)	Invalid login details	Error message	√
Registry (valid)	Valid personal and account details	Register the user	√
Registry (invalid)	Invalid personal and/or account details	Error message	√
Browse category	Category ID	Displays list of books within the category	√
Browse sub category	Category ID, sub category ID	Displays list of books within the sub category	√
Search (simple)	Simple search term	Displays list of books match the search term	√
Search (combination)	Mixed criteria	Displays list of books match the search criteria	√
Categories navigation	category	Displays a menu contains the sub categories belong to the parent category.	√
View user'profile	User ID	Display user's personal details.	√
View statisticl data	category	Displays a graph represent the user's preferences in the provided category.	√

Table 9.33: Integration Testing- Integrate Database with Front End increment

Add learning module increment			
System operation	**Input**	**Output**	**Expected Result**
Rate Book (insert)	Rating , user	1. Insert the rate to the database along with the provided user ID, book. 2. Update the provided user's profile.	√
Rate Book (update)	Rating , user	Update the user's rate for the provided book.	√
Add to favourite	book , user	1. Insert the book to the database along with the provided user ID. 2. Update the provided user's profile.	√
Add to shopping cart	book , user	1. Insert the book to the provided user's shopping cart. 2. Update the provided user's profile.	√
Browse a book	user	1. Displays book details 2. Update the provided user's profile.	√
Add to owned books	Book, user	1. Insert the book to the database along with the provided user ID. 2. Update the provided user's profile.	√
Fill knowledge based form	List of categories, user	Update the provided user's profile based on the provided categories.	√
View rated books	-	Displays books rated by user	√

Table 9.34: Integration Testing - Add learning module increment (1)

Add learning module increment			
System operation	**Input**	**Output**	**Expected Result**
View owned books	-	Displays user's owned books	√
View favourite books	-	Displays user's favourite books	√
View user's browsing history	-	Displays books in user's browsing history	√

Table 9.35: Integration Testing - Add learning module increment (2)

Add recommendation module increment			
System operation	**Input**	**Output**	**Expected Result**
Content based approach	Algorithm parameters	Displays list of recommended books	√
Content based approach (user did not evaluate enough books in the system)	Algorithm parameters	Error message	√

Table 9.36: Integration Testing - Add recommendation module increment (1)

Add recommendation module increment			
System operation	**Input**	**Output**	**Expected Result**
Collaborative filtering with person correlation algorithm(> 10 votes by active user)	Algorithm parameters	Displays list of recommended books	√
Collaborative filtering with person correlation algorithm(< 10 votes by active user)	Algorithm parameters	Error message	√
Collaborative filtering with MSD algorithm(> 10 votes by active user)	Algorithm parameters	Displays list of recommended books	√
Collaborative filtering with MSD algorithm(< 10 votes by active user)	Algorithm parameters	Error message	√
Hybrid approach(> 10 votes by active user)	Algorithm parameters	Displays list of recommended books	√

Table 9.37: Integration Testing - Add recommendation module increment (2)

Add recommendation module increment			
System operation	**Input**	**Output**	**Expected Result**
Hybrid approach (< 10 votes by active user)	Algorithm parameters	Error message	√
Browse a book	Book ID	Displays non-personalised recommendations.	√
Delete rated book	Book ID, user ID	1. Delete the provided book form the user's rated books list in the database. 2. Update the user's profile.	√
Delete favourite book	Book ID, user ID	1. Delete the provided book form the user's favourite books list in the database. 2. Update the user's profile.	√
Delete owned book	Book ID, user ID	1. Delete the provided book form the user's owned books list in the database. 2. Update the user's profile.	√
Delete form user's browsing history	User ID, book ID	1. Delete the provided book form the user's browsing history in the database. 2. Update the user's profile.	√

Table 9.38: Integration Testing - Add recommendation module increment (3)

9.13 Appendix M : Project Plan

ID	Task Name	Duration	Start	Finish
1	Research and Investigation	49.13 days?	Wed 07/05/08	Fri 23/05/08
2	Research Milestone	2 days	Sat 23/08/08	Sun 24/08/08
3	**Requirement Engineering**	**21 days?**	**Fri 23/05/08**	**Fri 30/05/08**
4	Feasibility Study	2 days?	Mon 26/05/08	Tue 27/05/08
5	Requirement Analysis	4.13 days?	Wed 28/05/08	Thu 29/05/08
6	Requirement Document	1.13 days?	Fri 30/05/08	Fri 30/05/08
7	Requirement Milestone	0 days	Fri 23/05/08	Fri 23/05/08
8	**System Design**	**49.13 days?**	**Sun 01/06/08**	**Tue 17/06/08**
9	Architecture Design	1 day?	Sun 01/06/08	Sun 01/06/08
10	Front End Increment	7.13 days?	Mon 02/06/08	Wed 04/06/08
11	Learning module Increment	13.13 days?	Thu 05/06/08	Mon 09/06/08
12	Recommendation Module incremer	16.13 days?	Tue 10/06/08	Sun 15/06/08
13	Database Increment	1 day?	Tue 17/06/08	Tue 17/06/08
14	Design Milestone	1 day	Tue 17/06/08	Tue 17/06/08
15	**Implementation**	**154.13 days?**	**Wed 18/06/08**	**Fri 08/08/08**
16	**Database Increment**	**10.13 days?**	**Wed 18/06/08**	**Sat 21/06/08**
17	Development	4.13 days?	Wed 18/06/08	Thu 19/06/08
18	Connection	1 day?	Thu 19/06/08	Thu 19/06/08
19	Data Extraction	7.13 days?	Thu 19/06/08	Sat 21/06/08
20	Front End Increment	19.13 days?	Sat 21/06/08	Sat 28/06/08
21	Learning Module Increment	19.13 days?	Mon 30/06/08	Sun 06/07/08
22	Integrate Increments	1 day?	Sun 06/07/08	Mon 07/07/08
23	**Recommendation Module Incre**	**79.13 days?**	**Tue 08/07/08**	**Sun 03/08/08**
24	Content-based Approach	16.13 days?	Tue 08/07/08	Sun 13/07/08
25	Collaborative Approach	28.13 days?	Mon 14/07/08	Wed 23/07/08
26	Hybrid Approach	1 day?	Thu 31/07/08	Thu 31/07/08
27	Recommendation Module Mile	10.13 days?	Thu 31/07/08	Sun 03/08/08
28	Integrate Increments	1 day?	Mon 04/08/08	Mon 04/08/08
29	Produce Final System	10.13 days?	Mon 04/08/08	Thu 07/08/08
30	Implementation Milestone	14 days	Mon 04/08/08	Fri 08/08/08
31	**Testing**	**151.13 days?**	**Sat 28/06/08**	**Sun 17/08/08**
32	Unit Testing	118.13 days?	Sat 28/06/08	Wed 06/08/08
33	Integration Testing	94.13 days?	Mon 07/07/08	Thu 07/08/08
34	System Testing	4.13 days?	Mon 11/08/08	Tue 12/08/08
35	Beta Test	19.13 days?	Mon 11/08/08	Sun 17/08/08
36	**Evaluation**	**76.13 days?**	**Mon 28/07/08**	**Fri 22/08/08**
37	Research and Investigation	19.13 days?	Mon 28/07/08	Sun 03/08/08
38	Design	1 day?	Tue 05/08/08	Tue 05/08/08
39	Implemenation	1 day?	Fri 15/08/08	Fri 15/08/08
40	Live User Experiment	19.13 days?	Tue 12/08/08	Mon 18/08/08
41	Evaluation Document	6 days?	Mon 18/08/08	Wed 20/08/08
42	Evalaution Milestone	7.13 days?	Wed 20/08/08	Fri 22/08/08
43	Complete Report	247.13 days?	Sun 01/06/08	Fri 22/08/08
44	Report Milestone	7.13 days?	Fri 22/08/08	Sun 24/08/08
45	Presentation	19.13 days?	Wed 27/08/08	Tue 02/09/08

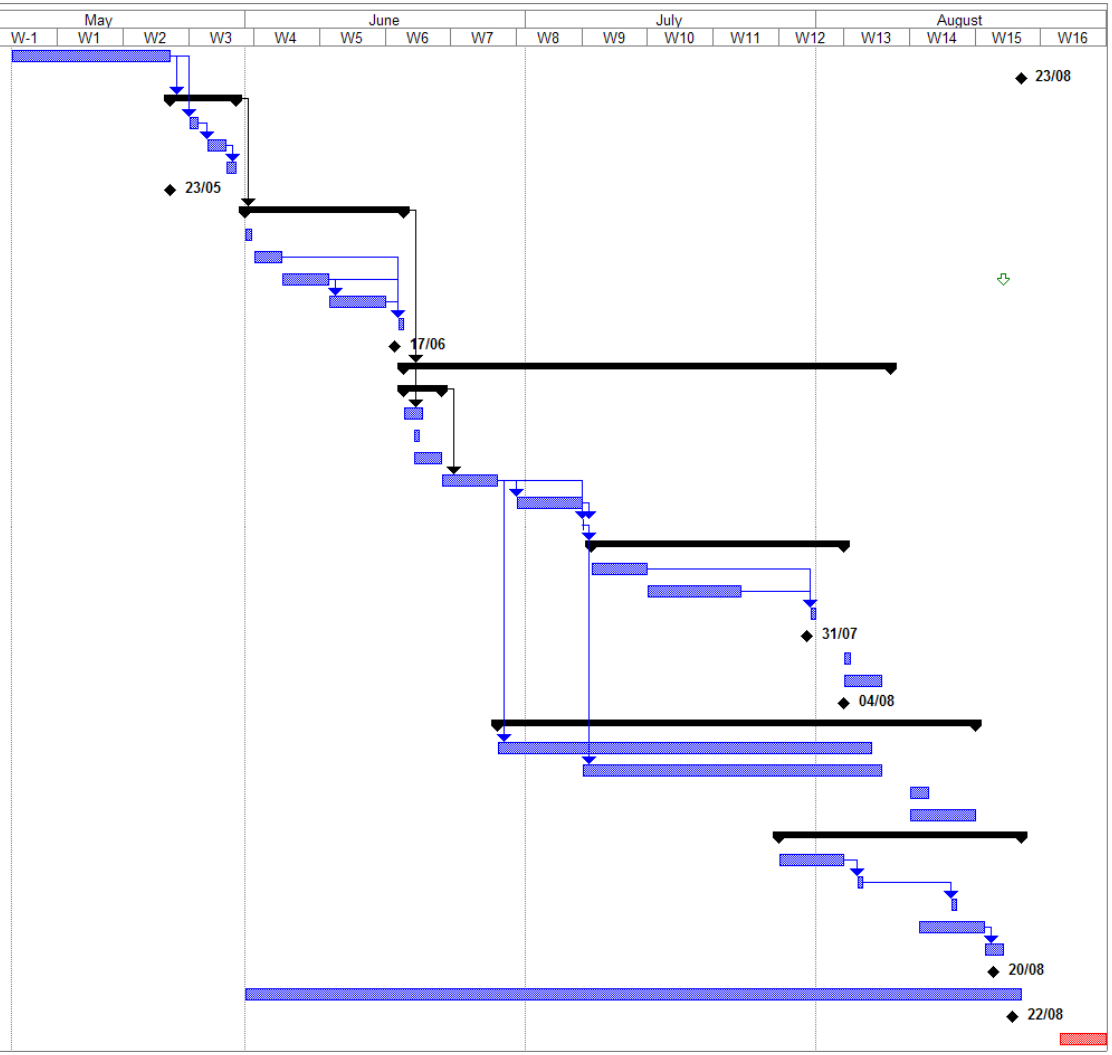


Figure 9.29: Project Plan