Evaluation form – Progress Reports

V1.06 Feb. 2004

Project ID	203252	Project Title	An Interactive Recommender System
Student Name	Bernard Ma	Supervisor	Prof. R. Zemel
Section #	6	Coordinator	Beresford

Presentation	Copy graded: ❐ electronic ❐ paper	❐ comment summary in paper
Coordinators Signature:		Grade /10 (each report worth 2.5%)

Technical Evaluation	**Total Mark**		**Suggested Mark**		**Notes**
Progress	10				• following game plan? (milestones accomplished; milestones missed & why)
Organization	10				• use of human resources? • use of non-human resources? • efficiency of efforts? • preparation & foresight?
Method	15				• are the modules / steps sufficiently tested when 'done'? • do these tests have good structure? • have the interaction specifications of the parts been properly developed?
Decision making	10				• suitable response to difficulties? • appropriate change of course when encounter obstacle or new development? • modification of process as necessary?
Creativity / Complexity / Effort	35	C/C ____ Effort x ____ x 35			Creativity and Complexity rom charts. Scores for creativity, complexity and effort are multiplied by maximum score / 1000 to get the grade.
TOTAL	80				(report #1 worth 7.5 & report #2 worth 12.5% of the student's final grade)
Supervisor	❐ Accept suggested technical mark ❐ Change technical mark (reasoning attached in separate sheet, marks in blank column above)				
Supervisor's Signature:					

Note to students: There is a design award, the Aloha award, that you might wish to apply for. Please check the course website for instructions on what to do to be considered for this award.

Page 2 of Progress Report Evaluation Sheet

Project ID		Project Title			
Student Name			Supervisor		
Section #		Coordinator/Administrator			

Averages for this section: Presentation ______

Technical ______

Return this evaluation sheet as described below to
❒ Coordinator
❒ _______________

Supervisors:
Please review the progress reports for the students in your group(s). The terms are further explained on the course website using the menu at the top of the website to get to the links on the page at
Student Information | Deliverables/Evaluation | Individual Progress Reports
[The course website is at http://courses.ece.utoronto.ca/ece496y1y/.]
In particular,

- the 'effort' assessment is generally .7 to 1.2, but normally close to 1.0. Follow the link to 'Marking Terms'
- the creativity/complexity mark comes from a graph. Follow the link to 'Mark Matrix'.

You may choose to accept the technical mark on the first page, or to change the marks. If you choose to change the mark, please enter the changed mark in each category on the first page and write a note of justification for the change. The student final grade will be determined after consultation.

Please do not return the reports to the students until you receive notification to do so.
Please return a copy of the first evaluation sheet to the Coordinator with a justification if necessary within a week.

The average marks for the section are shown above. There will be differences between the marks in each section. Marks between sections will be normalized after the final reports are marked (about mid-April). The students are already aware of this.

Supervisors Comments	
Supervisor's Signature:	

The Edward S. Rogers Sr. Department of Electrical and Computer Engineering University of Toronto	
ECE496Y Design Project Course Individual Progress Report	
Title: An Interactive Recommender System	
Project I.D.#: 2003252	
Prepared by:	Bernard Ma – bernard.ma@utoronto.ca
Supervisor:	Prof. R. Zemel
Section #:	6
Section Administrator:	D. Beresford
Date:	Monday, February 23, 2003

Executive Summary

E-commerce sites often employ a feature known as recommender systems. The purpose of this feature is to intelligently recommend products customers are likely to be interested in. Ideally, this acts as an incentive for customers to continue to access the site. Unfortunately, recommender systems often make poor recommendations, often turning off the user from their online shopping experience.

Many recommender systems employ the Collaborative Filtering (CF) methodology in making their recommendations. The main problem with CF resulting in inaccurate recommendations is known as the "new user problem". Active Collaborative Filtering (ACF) is a recently developed methodology which seeks to solve the "new user problem." The purpose of this project is to implement the ACF methodology into a full-scale system of high complexity. This system will test the ACF methodology (and other recommender algorithms) on real users by making recommendations to them on music. After this is complete, we will analyze and attempt to explain the results of the testing.

Currently, the Web Interface is fully functioning, complete with: Registration and User Recognition and mp3 playback (for Windows users). Integration with the Database Module and Application Module has also been completed. We are currently using the system to perform the analysis phase of the project. The project is currently progressing on schedule.

Table of Contents

1. Introduction

In the retail industry, e-commerce has been gaining prominence over the last decade [2]. As competition over market share grows more fierce, companies continue to search for ways to distinguish themselves from the competition. A main feature provided by many of these online sites is the recommender system. A recommender system intelligently makes suggestions on products it believes the consumer will have an interest in purchasing. A recommender system determines what products to recommend based on the ratings the consumer makes on other products. This is a form of target marketing

Unfortunately, these recommendations made for the consumer are often inaccurate. As a result, these inaccurate recommendations can have an adverse affect, turning off the consumer from the site instead of enticing him/her. Many of these recommender systems employ the Collaborative Filtering (CF) methodology in making recommendations for the customer. The CF methodology simply takes in a set of user ratings as input, and uses this information to make recommendations. The main problem with this technique which leads the inaccurate recommendations is known as "The New User" problem. In order for the CF methodology to provide accurate recommendations to the customer, many user ratings from the customer is required. However, when a user logs onto an e-commerce site for the first time, little information is known about the new user. As a result, poor recommendations are made.

A new approach known as Active Collaborative Filtering (ACF) has been developed by Professor Craig Boutilier, Professor Richard Zemel and graduate student Benjamin Marlin at the University of Toronto in an attempt to solve this problem. The main idea behind ACF is to gather as much useful information about the user as possible through querying the user for ratings on various products. To minimize the number of queries made, the ACF approach utilizes the probability model known as EVOI (Expected Value Of Information) to query users on products that will yield the most information about the user [4]. From here, probability models are used to intelligently calculate recommendations for the user. In effect, this provides accurate recommendations for the user while querying them a minimum number of times.

To date, the ACF approach has only been tested in laboratories. The purpose of the design project will be to implement ACF on a full scale system in order to test this ACF approach on real users. The system will make recommendations on music clips. Other recommender approaches will also be implemented into the system and tested on users. Results of these tests will be analyzed, and will likely suggest ways of improving on the ACF implementation. The project objectives are as follows:

1) Implement ACF in Java
2) Create a web-based user friendly interface using Java Servlets
3) Create an easy to access database of songs, samples and ratings using MySQL
4) Transmit data between the above 3 modules of the system
5) Analyze a select group of recommender algorithms after running tests on our system

My focus for the project will be to provide a non-static, user-adaptable Web Interface representing the front-end of the recommender system. This Web Interface must query the user for ratings, employing the ACF methodology. This means, querying the user for ratings on the song (optimal query) which will yield the most information about the user. I will also be responsible for integration of the Web Interface with the Application (back-end) and Database (guts). This report will discuss the progress made in these areas.

2. Progress

This Section will describe the progress made to date with respect to the milestones I am responsible for delivering. They are as follows:

- Web Server / GUI Setup
- Integration of Web Server and Application
- Integration of Web Server and Database
- Design Fair Poster Presentation

The Milestones are numbers 4, 7, 6 and 13 (respectively) and can be found in Appendix O. The original milestone chart can be found in Appendix N. Progress on the *Design Fair Poster Presentation* will not be discussed as it has not yet begun. Since the last progress report, the responsibility of the milestone: "Integration of Web Server and Database" was transferred to myself from Andrew Yeung. This was a result of the nature of the integration process as it was more of an extension from the Web Server module to the Database module rather than from the Database module to the Web Server module. It also required an intimate understanding of the inner workings of the Web Server, as code would be modified in various locations within the Servlets.

During the integration phase, it was discovered that the communication delay between the Web Server and Database module was instantaneous, thus making the testing of the speed trivial for this milestone (milestone #6). The milestone was rewritten accordingly. The testing for both integration milestones (milestone #6 and #7) were re-written with the focus on the correctness of the data exchanged between modules. Dates were updated as a result of the Web Server / GUI Setup milestone delayed. A detailed account of the progress for the milestones is provided below:

2.1 Web Server / GUI Setup

2.1.1 Milestone Overview

Milestone & Description A user friendly and informative interface will be designed and implemented. The GUI will provide features such as information about the music to be rated (mp3 sound clips, artist information etc.). Due: December 13, 2004
Responsibility: Bernard Ma
Status at start of reporting period: 80% complete. Most of the framework of the Web GUI had been implemented on an Apache/Tomcat Web Server and was in the process of being debugged.
Status at end of reporting period: Completed December 13, 2004

2.1.2 Actions

The layout of code was designed in a modularized fashion. "Dummy Classes" were written to mimic the Application module and the Database module. A more detailed discussion of can be found in Appendix G.

During implementation, a number of problems arose:

- GUI (i.e. tables in HTML) not being displayed as envisioned
- Session Data not being stored properly

Changes were made in the code to rectify these problems. A more detailed discussion can be found in Appendix H.

In order to complete the milestone, the feature: User Login/User Recognition needed to be implemented. In order to implement Login and User Recognition, some extensions needed to be implemented in the *change of state* design of the system to handle this additional complexity. Figure 1 illustrates the changes of state in the system (The previous version was only made up of the "Recommender System" component).

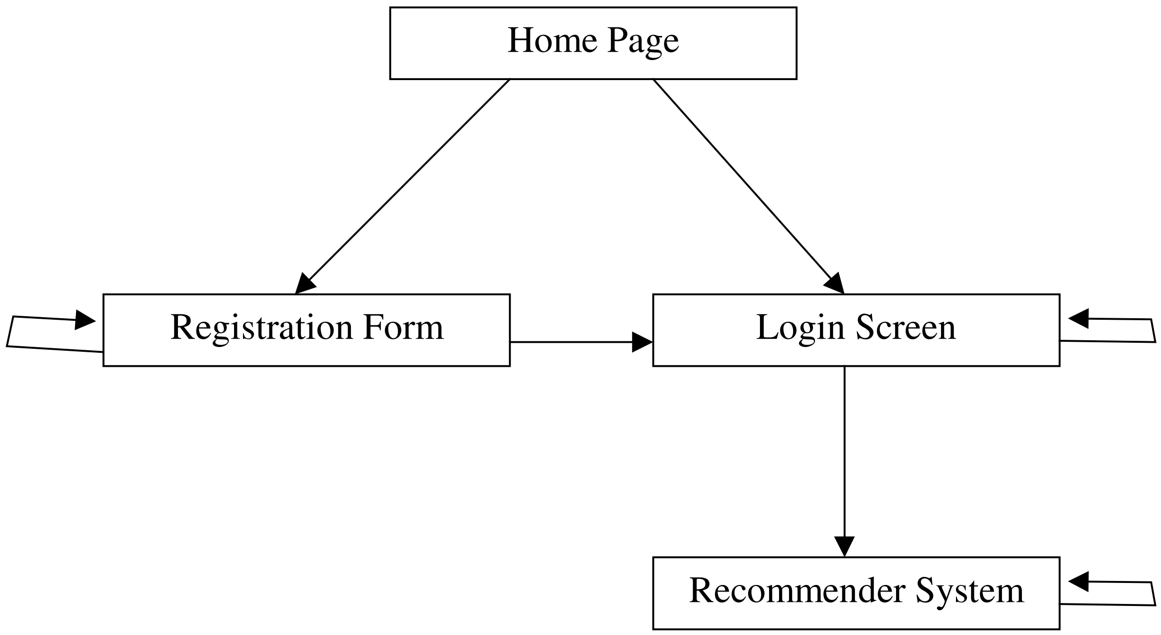


Fig 1: Changes of State for the System

The implementation of the Login/User Recognition was accomplished through coding in Servlets. A more detailed discussion can be found in Appendix I. The code can be found in Appendix D (Login.java) and Appendix E (Register.java).

2.1.3 Decisions

To meet the goal of a user friendly user interface, three decisions were made. Firstly, the code was designed in a modularized fashion. The Servlets functioned as the "middleman" interfacing with the User, Application Module and Database Module. Figure 2 illustrates this relationship.

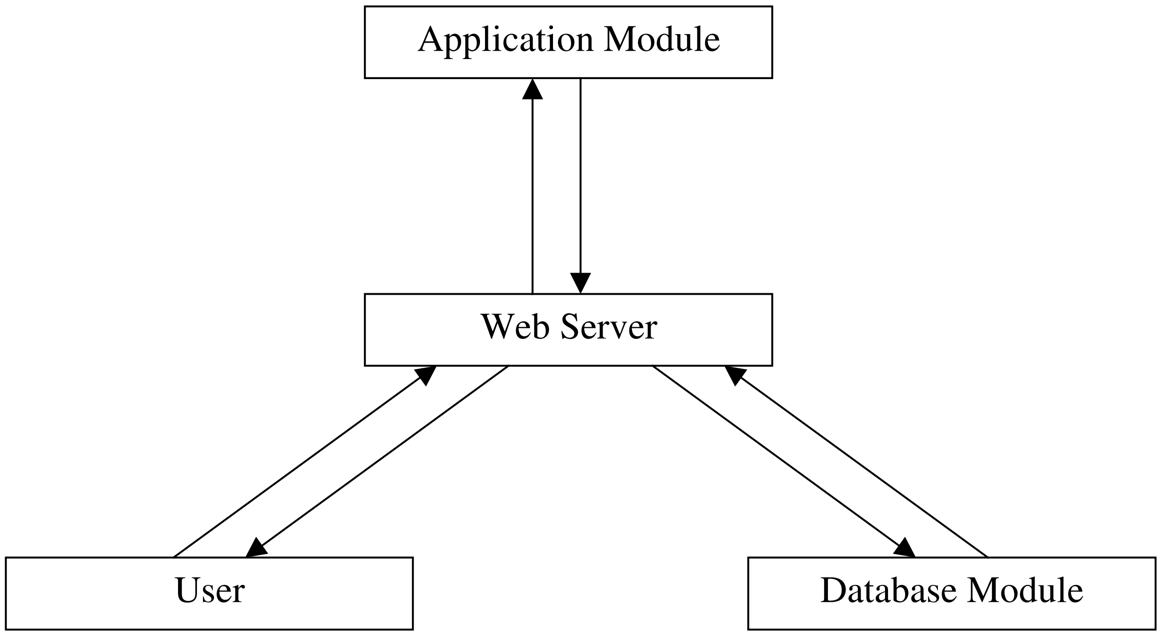


Fig 2: Relationship between Modules

The Application and Database Modules are the sources for providing song information to the Web Server (i.e. artist information, recommendation etc.). Modularity was taken into account in the design of the system such that the number of modifications to the code in the Servlets would be minimized during the integration phases. Also, the communication between modules would be well defined, minimizing the likelihood of errors during the exchange of data. The dummy classes would simply be replaced will "real" classes which communicate with the Application Module and Database Module, while the code in the Servlet (Appendix C: Optimal.java) would continue passing the same arguments as it did to the dummy classes, with some minor adjustments as necessary. This allows for concurrent implementation of the modules.

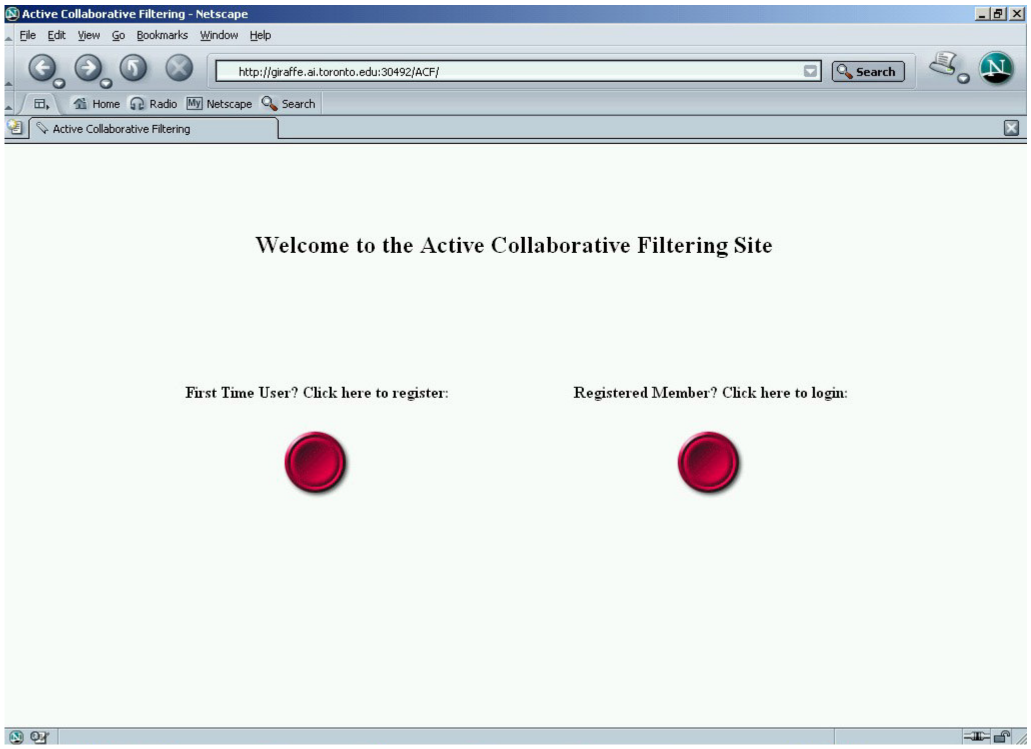
The next decision was made to complete the objective of implementing the Web Server. The decision was made to use Servlets to code the Login Screen and Registration

Form. Both of these GUI's needed to be dynamic in order to interact with the user (i.e. User logs on with an incorrect password, Login screen prompts the user to re-enter password).

2.1.4 Testing & Verification Progress

A test was performed to verify that the Session Data was being stored and retrieved correctly during a user's session. Whenever Session Data was being stored and retrieved (i.e. ratings, user id), the resulting data retrieved was displayed onto the screen. This test was performed for numerous iterations and scenarios. After a number of iterations of the above tests, we could confirm the Session Data was being stored and retrieved correctly.

The time between a user submitting a rating and receiving the Next Optimal Query/Recommendations was almost instantaneous. As a result, no speed tests were necessary to verify the system was performing at an acceptable speed. Finally, Usability Tests were performed on the overall usability and esthetics of the system. User's suggestions were collected pertaining to the layout of information. Suggestions were compiled, and certain improvements were implemented. The current layout and design can be seen in Figures 3, 4, 5 and 6.

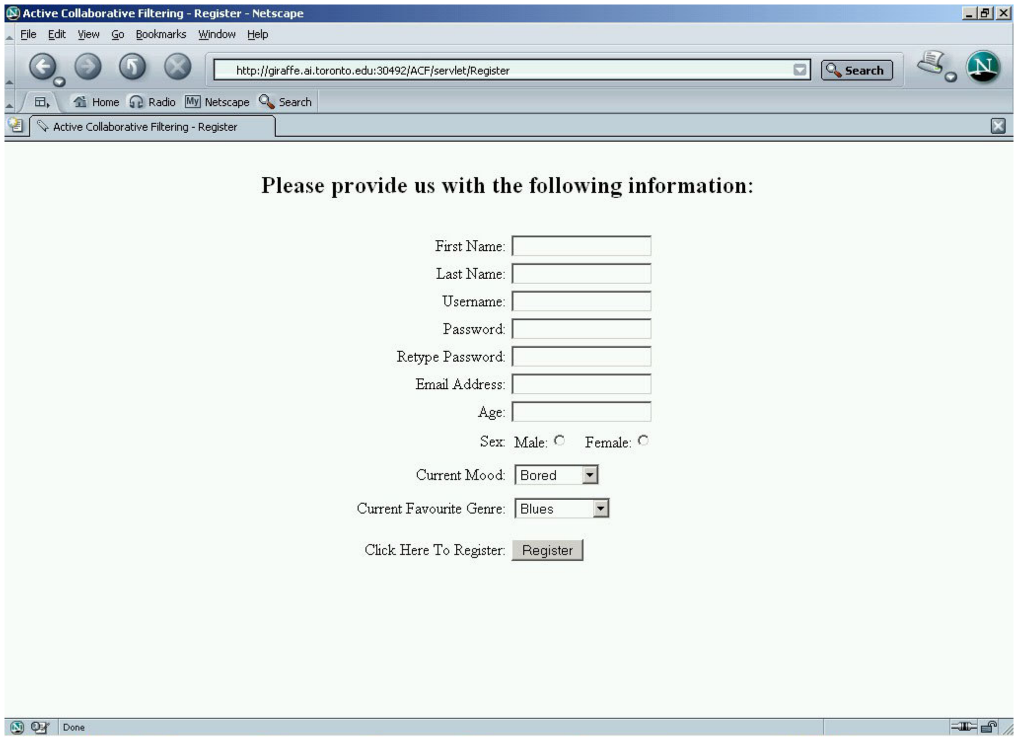

Active Collaborative Filtering - Netscape
File Edit View Go Bookmarks Window Help
http://giraffe.ai.toronto.edu:30492/ACF/
Search
Home Radio Netscape Search
Active Collaborative Filtering

Welcome to the Active Collaborative Filtering Site

First Time User? Click here to register:

Registered Member? Click here to login:

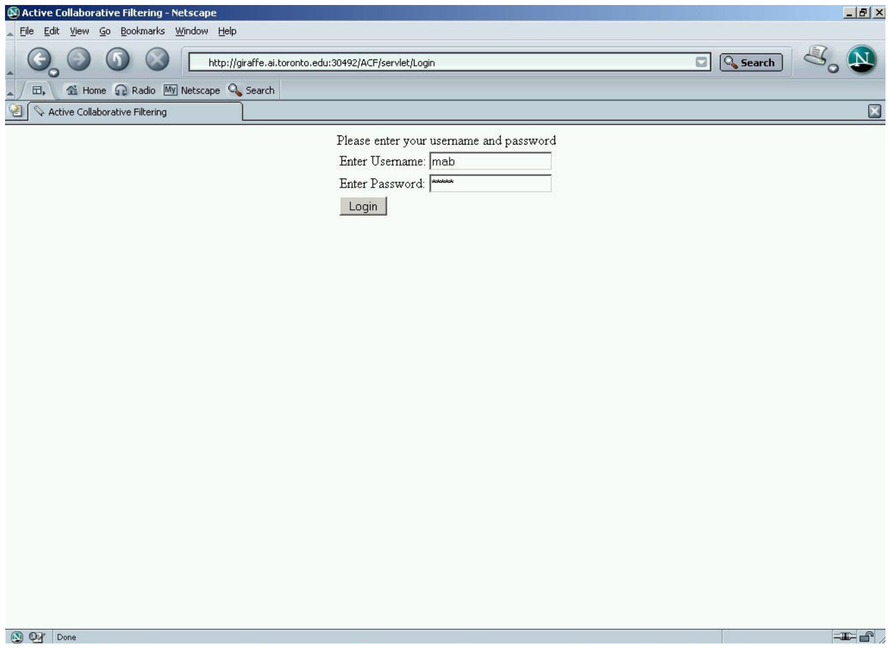
Fig 3: Home Page


Active Collaborative Filtering - Register - Netscape
File Edit View Go Bookmarks Window Help
http://giraffe.ai.toronto.edu:30492/ACF/servlet/Register
Search
Home Radio Netscape Search
Active Collaborative Filtering - Register
Done

Please provide us with the following information:

First Name:	
Last Name:	
Username:	
Password:	
Retype Password:	
Email Address:	
Age:	
Sex:	Male: ○ Female: ○
Current Mood:	Bored
Current Favourite Genre:	Blues
Click Here To Register:	Register

Fig 4: Registration Form

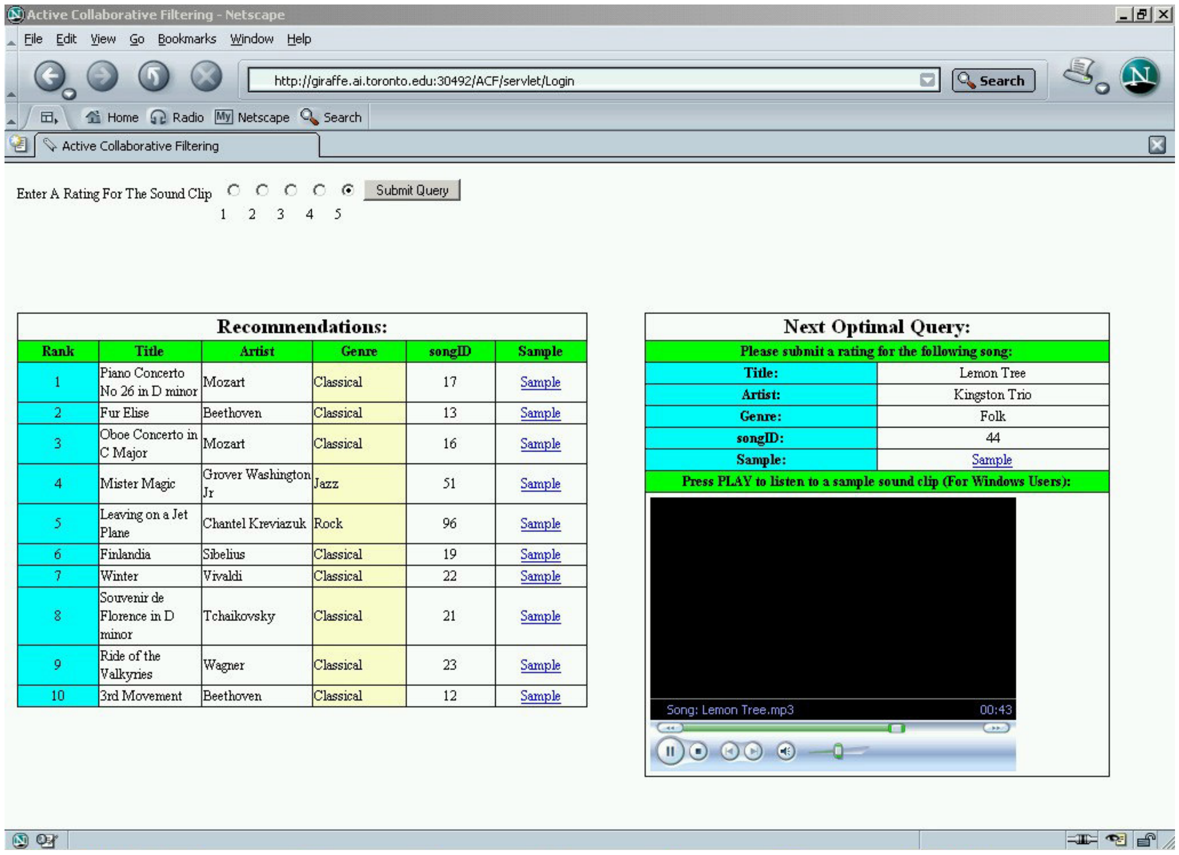

Active Collaborative Filtering - Netscape
File Edit View Go Bookmarks Window Help
http://giraffe.ai.toronto.edu:30492/ACF/servlet/Login
Search
Home Radio Netscape Search
Active Collaborative Filtering
Done

Please enter your username and password

Enter Username:	mab
Enter Password:	*****

Login

Fig 5: Login Screen


Active Collaborative Filtering - Netscape
File Edit View Go Bookmarks Window Help
http://giraffe.ai.toronto.edu:30492/ACF/servlet/Login
Search
Home Radio Netscape Search
Active Collaborative Filtering
Song: Lemon Tree.mp3
00:43

Enter A Rating For The Sound Clip 1 2 3 4 [✓ 5] Submit Query

Recommendations:

Rank	Title	Artist	Genre	songID	Sample
1	Piano Concerto No 26 in D minor	Mozart	Classical	17	Sample
2	Fur Elise	Beethoven	Classical	13	Sample
3	Oboe Concerto in C Major	Mozart	Classical	16	Sample
4	Mister Magic	Grover Washington Jr	Jazz	51	Sample
5	Leaving on a Jet Plane	Chantel Kreviazuk	Rock	96	Sample
6	Finlandia	Sibelius	Classical	19	Sample
7	Winter	Vivaldi	Classical	22	Sample
8	Souvenir de Florence in D minor	Tchaikovsky	Classical	21	Sample
9	Ride of the Valkyries	Wagner	Classical	23	Sample
10	3rd Movement	Beethoven	Classical	12	Sample

Next Optimal Query:

Please submit a rating for the following song:

Title:	Lemon Tree
Artist:	Kingston Trio
Genre:	Folk
songID:	44
Sample:	Sample

Press PLAY to listen to a sample sound clip (For Windows Users):

Fig 6: Recommender System

2.2 Integration of Web Server and Application

2.2.1 Milestone Overview

Milestone & Description In order to maximize the effectiveness of the system, the Web Server must interact quickly will the application. Integration will focus on the speed of communication between the modules. Extensive testing must be done to demonstrate the correctness of the data exchanged. Modifications to the modules will be made if necessary. Due: January 24, 2004
Responsibility: Bernard Ma
Status at start of reporting period: Not started.
Status at end of reporting period: Completed January 17, 2004

2.2.2 Actions

In order to meet the objective of quick interaction between the Web Server and Application module, the following two actions were performed:

1. The decision was made for the *Prediction Matrix* to be rebuilt every time the user submits a new user rating. A detailed discussion on the Prediction Matrix can be found in Appendix J. The Servlets were coded to facilitate this decision. The current list of ratings was now sent to the Application at every rating submission by the user and the Dummy Application module was now coded as a static class. The code for Optimal.java can be found in Appendix C.
2. Servlets were coded with the user ratings defined as an array of "int's". Please see Appendix K for a detailed discussion about the data structures.

By defining the interface while the modules were implemented concurrently, the integration was flawless. The dummy class was simply replaced with the "real" Application class.

2.2.3 Decisions

The *Web Server / GUI Setup* milestone was implemented concurrently with the *Implement ACF Methodology in Java* milestone. Throughout this time period, many discussions between Cavan Yie and myself took place pertaining to:

1. Whether to keep the Prediction Matrix "alive" throughout a user session
2. The type of Data structure to use for user ratings

These two factors were key in meeting the objective of quick interaction between the Web Server and Application module.

For the issue of the prediction matrix, our discussions focused on whether this Prediction Matrix would be kept "alive" throughout the duration of the user's session (option 1), or if it would be rebuilt throughout the user's session each time a new rating is submitted (option 2). Both options had their pros and cons. See Appendix K for a detailed analysis of the two options. In the end, option 2 was determined to be superior.

The second factor in determining the speed of the system was the Data structure of the user ratings sent to the Application. The decision was made to define the data structure as an array of "int's" rather than its original structure of a "Hashtable". The runtime of the ACF methodology (in the Application Module) using the Hashtable structure was 1-4 minutes. Having the user wait several minutes between rating submissions was unacceptable. Changing the data structure to an array of "int's" improved the runtime of the ACF methodology to 1-4 seconds. Please see Appendix L for a detailed discussion about the data structures.

<u>2.2.4 Testing & Verification Progress</u>

The following tests were performed to test that the correctness of data exchanged between the Web Server and Application Module was correct:

Whitebox Testing was performed on the Application in isolation (ACF.java) to ensure that the module would return proper results for different sets of ratings. A main class was written, calling the methods required, and outputting the return data to the terminal. 5 test cases were defined for a set of 124 songs: sending 0 ratings, 35 ratings, 70 ratings, 105 ratings and 124 ratings. The same data was returned consistently after running each case a number of times. The Application passed this test.

Blackbox Testing was performed to test that the connection between the modules was correct. The same ratings used for the Whitebox testing case were manually input through the Web GUI. The results (recommendations and Next Optimal Query) matched the results of the 5 test cases used for the Whitebox testing. The communication between the Web Server and Application passed this test.

2.3 Integration of Web Server and Database

2.3.1 Milestone Overview

Milestone & Description Interface for modular database component will have a focus on easy access by the web server. Extensive testing must be done to demonstrate the correctness of data stored/retrieved to/from the database. Modifications to the modules will be made if necessary. Due: January 24, 2004
Responsibility: Bernard Ma
Status at start of reporting period: Not started
Status at end of reporting period: Completed January 24, 2004

2.3.2 Actions

To meet the objective of the Web Server having an easy access of the Database, the following actions were performed:

1. The JDBC driver was downloaded and integrated into the Web Server from the MySQL homepage (www.mysql.com)
2. The sample code: *PointBaseExample.java* was used as a template to code: *dbConnect.java* [7]

The dummy code representing the database was then replaced with: *dbConnect.java* (Appendix F). Please see Appendix M for a detailed description of the communication process between the Web Server and Database coded in dbConnect.java.

2.3.3 Decisions

To meet the objective of the Web Server having an easy access of the Database, the following decisions were made:

1. Andrew Yeung had informed me that a *JDBC driver* needed to be installed on the Web Server as it was required for an Apache/Tomcat Web Server to access a MySQL database. The decision to use the JDBC driver was straight forward.
2. Through my prior research into Java Servlets from the University of Toronto course CSC309 website [7], I had encountered sample code of a Java Servlet interacting with a *PointBase* database: *PointBaseExample.java*. The decision was made to use this sample code as a template was made after testing out certain commands in the code and discovering that they were relevant for accessing a *MySQL* database.

2.3.4 Testing & Verification Progress

To meet the objective of having correctness of data exchanged between the Web Server and Database Module, the following tests were performed:

Whitebox Testing was performed to ensure the correct data was being read from database. A main class was written in dbConnect.java calling on the "read" methods in dbConnect.java, sending it a specific parameter. The return data was observed and cross-checked with the Database to verify correctness. An exhaustive number of test cases were run with the program passing each time. This test was repeated to ensure correctness in writing to the database by calling on the "write" methods in dbConnect.java.

Blackbox Testing was performed to test that the connection between the Servlet and the *dbConnect.java* is correct. A *read* test was performed by outputting the data retrieved from the "read" methods. An exhaustive number of test cases were run with the program passing each time. This test was repeated for writing to the database by calling on the "write" methods in dbConnect.java. An exhaustive number of test cases were run with the program passing each time.

3 Conclusion

To date, the Web Server is fully functioning, complete with: Registration and User Recognition and mp3 playback (for Windows users). The modularity of the code allowed for smooth integration with the Database Module and Application Module. All components have been tested are considered correct. The web site is hosted at: "http://giraffe.ai.toronto.edu:30492/ACF".

References

[1] B. Marlin. Active Collaborative Filtering with Bayes. Unpublished.

http://www.cs.toronto.edu/~marlin/research/research.shtml. 2002.

[2] Cox, B. 2003. "E-commerce News: E-commerce Industry Soaring." E-commerce-Guide Website.

http://ecommerce.internet.com/news/news/article/0,,10375_1585731,00.html. Site accessed 24/9/03.

[3] Craig Boutilier, Richard S. Zemel, and Benjamin Marlin. Active Collaborative Filtering. In Proc. of the 19th

Conference on Uncertainty in Artificial Intelligence. 2002.

[4] Craig Boutilier and Richard S. Zemel. Online queries for collaborative filtering. In AI-Stat, 2003.

[5] J. Ben Schafer, Joseph Konstan, John Riedl. Recommender Systems in E-Commerce. In Proc. of the 1st AC

conference on Electronic commerce 1999.

[6] L. Guernsey. "Making Intelligence a Bit Less Artificial." The New York Times.

http://www.cs.toronto.edu/~marlin/library/nyt_rec.pdf. Site accessed 01/8/2003.

[7] D. Penny. "CSC309F Lectures"

http://www.cs.toronto.edu/~penny/teaching/csc309/lectures/ Site accessed 30/10/03

[8] S. Zeiger. "Servlet Essentials"

http://www.novocode.com/doc/servlet-essentials/ Site accessed 10/11/03

[9] Kurniawan. B. "How Servlet Containers Work"

http://java.sun.com/products/servlet/docs.html Site accessed 05/11/03

Appendix A – index.html

```
<!-- file: index.html -->
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
<html>
<head>
<title>Active Collaborative Filtering</title>

<link rel="stylesheet" type="text/css" href="mystyle.css" />
</head>

        <!-- 3 frames in this document, "top" is for querying User, main is for the displaying
Song Info of Song to be rated,
        topTen is the left.  Displays Top 10 Recommendations -->
        <frameset rows="120,*,120" frameborder="0"  >
                <frame name="topFrame" noresize="noresize" src="top.html"> <!--
Provides querying facility for user-->
                <frameset cols="150,*" frameborder="0">
                        <frame name="topTen" noresize="noresize" src="topTen.html">
<!--Displays the top ten items...not yet implemented-->
                        <frame name="mainFrame" noresize="noresize"
src="../servlet/Optimal">   <!--Displays infor of next Optimal Query-->
                </frameset>
        </frameset>
</html>
```

Appendix B – top.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
<html>
<head>
<link rel="stylesheet" type="text/css" href="../mystyle.css" />
</head>
<body scroll = "no">
	<table>
	<tr><td>
		<form method="get" action="http://localhost:8080/servlet/Optimal"
target="mainFrame">
		Enter A Rating For The Sound Clip
	</td><td>
		<table>
			<tr>
				<td> <input type=radio name=rating value=1> </td>
				<td> <input type=radio name=rating value=2> </td>
				<td> <input type=radio name=rating value=3 checked> </td>
				<td> <input type=radio name=rating value=4> </td>
				<td> <input type=radio name=rating value=5> </td>
				<td> <input type=submit> </td>
			</tr>
			<tr>
				<td align="middle"> 1 </td>
				<td align="middle"> 2 </td>
				<td align="middle"> 3 </td>
				<td align="middle"> 4 </td>
				<td align="middle"> 5 </td>
				<td></td>
			</tr>
		</table>
	</td></tr>
	</form>
</body>
</html>
```

<u>**Appendix C – Optimal.java**</u>

See Optimal.java

Appendix D – Login.java

See Login.java

<u>**Appendix E – Register.java**</u>

See Register.java

Appendix F – dbConnect.java

See dbConnect.java

<u>Appendix G – Dummy Classes</u>

"Dummy Classes" were written to mimic the Application module and the Database module. These classes would return the data expected from the actual Application and Database (in development at this time). Session Data was utilized in mimicking the Database, as ratings submitted by the user would be stored in Session Data rather than directly to the database (which will be implemented during the Integration phase, milestone #6).

<u>Appendix H – Problems in Web Server</u>

During implementation, a number of problems arose. Oftentimes the GUI, specifically the tables coded in HTML, would not be displayed as envisioned. My past experience with web development aided in rectifying these formatting issues. Another issue I faced at the time of the reporting period was *Session Data* not being stored properly in the Web Server. As stated previously in Progress Report #1, the data being stored in the Session Data did not always match the data submitted by the user query. After doing some tests, it was discovered that the data was being stored properly, but was not being retrieved properly. The advantage of using Servlets, is user interaction through data that is exchanged between the Client and Server

This Client-Server interaction is essential for e-commerce sites. Every time a user accesses a site, they are assigned a *Session ID* which represents the user throughout the session. The Session ID is sent to the server through an *HTTP Request* every time data exchange occurs between the Client and Server. As multiple users can access the site simultaneously, this Session ID is essential in determining which user the Session Data belongs to. If the HTTP request received by the Server is not handled correctly, the Session

ID will not be received by the Server and Session Data previously stored for that Session ID will not be retrieved. The code was modified so that the Servlet could properly service the HTTP request, receiving the Session ID properly.

Appendix I Communication Between Servlets

The implementation of the Login/User Recognition required the coding of a Login screen and a Registration form using Servlets. The Servlets: *Login.java* (Appendix E) and *Register.java* (Appendix F) were written to represent the Login Screen and Registration Form respectively. The coding was not too difficult, as the experience from coding *Optimal.java* provided me with a solid understanding of how Servlets work. At this stage I did not know how to pass data between the Servlets. i.e. When an existing user logs in, how do you pass the username to *Optimal.java* so that it can access the previous song ratings the user had submitted? Fortunately, I gained a thorough the understanding of Sessions through my previous debugging. It became apparent that data could be passed between Servlets by storing that data in Session Data which is accessible for any Servlet during the user's session. Therefore, the user id would be stored in session data.

Appendix J – Prediction Matrix

Based on the ratings submitted by the user (through the Web Interface), the Application Module would send back the *Next Optimal Query* and *List Of Recommendations* (represented by song ID's) to the Web Server. In order to compute the *Next Optimal Query* and *List Of Recommendations*, a *Prediction Matrix* needs to be built based on the ratings the user has made as the ratings from the survey conducted offline.

From this *Prediction Matrix*, EVOI (Estimated value of Information) for each item would be computed.

Appendix K – Keeping Prediction Matrix Alive

For the issue of the prediction matrix, our discussions focused on whether this Prediction Matrix would be kept "alive" throughout the duration of the user's session (option 1), or if it would be rebuilt throughout the user's session each time a new rating is submitted (option 2). Option 1 would save overhead required in rebuilding the prediction matrix. An instance of Cavan's class could be created and stored in session data, keeping the prediction matrix alive. However, problems would arise if the user were to log off, and log on again. The Prediction Matrix "dies" after the user's session ends when they log off. A more complicated communication process between the Web Interface and Application would be required as the Web Interface must indicate whether the Prediction Matrix will need to be re-built. Also, problems could occur as the Prediction Matrix can only be kept alive throughout the user's session. However, in the case of a "timeout", if a user does not submit a rating for an extended amount of time, the session may be discarded, "killing" the prediction matrix and as a result, killing the process. Option 2 makes for a more "clean" interface and also solves this "timeout" issues common for web applications. For example, a user could log on to the system and submit a rating well after the timeout and the system would still operate as correctly. The same Session ID is no longer a requirement of the system. Thus, the decision was made to not keep the Prediction matrix alive throughout the user session. Instead, the Prediction matrix would be rebuilt throughout the user's session each time a new rating is submitted.

Appendix L – Evolution of Data Structure

The data structure went through 3 stages of evolution. Originally, the data was defined as a counter (representing the number of ratings) and two Hashtables (representing the Rating corresponding songID), as I was of the understanding that the order the ratings submitted would affect the computation of the *EVOI* and in turn, affect the Next Optimal Query and the list of Recommendations.

When I was informed that the order of ratings submission does not play a factor in the Application, the data structure was modified to a single Hashtable with the index as the songID and the rating as the output. The advantage of using a Hashtable was twofold. First, the Hashtable could be stored in Session Data as an object (as the integration with the Database had yet taken place). Secondly, Hashtables are dynamic. As more ratings are submitted, the size of the Hashtable would increase with the number of ratings. The alternative to this would be allocating the memory of all 126 ratings before hand. Using Hashtables would save overhead in that the Application would not have to traverse through 126 data types including those for songs not yet rated.

When the Application was tested with this data structure, the runtime of the ACF methodology would take 1-4 minutes depending on the number of ratings submitted. It was determined that the overhead in extracting data from the Hashtable was too great. When I was informed the runtime of the ACF methodology still took 1-4 minutes, the decision was made to change the data structure again by predefining 126 memory allocations in an array of "int's". When the Application module was tested using this data structure, the time to run the ACF methodology would take 1-4 seconds depending on the number of ratings submitted. Thus, the 3rd and final modification to the data structure was implemented, using an array of int's.

Appendix M – Coding dbConnect.java

With the JDBC driver installed, and the PointBase example as a template, the dummy code representing the database was replaced by: *dbConnect.java* (Appendix D). dBConnect.java is a static java class with separate methods written to accommodate any Database interactions required. Instead of the user ratings being stored in Session Data, the ratings would now be stored directly into the database.

Three categories of methods were coded:

1. Methods to "connect" and "disconnect" to and from a particular database. This must be done before any queries can be run on a particular table.
2. Methods to query a particular table. This includes reading or writing from or to a particular table. After a connection is established above, a simple SQL command is sent to the Database based on information provided from *Optimal.java*. (i.e. Optimal.java required the Title, Artist and Genre for a Particular songID. This method will query the database for this songID and return the desired information).
3. Methods to verify information. These methods were implemented with respect to the Login and Registration Servlets to handle corner cases. (i.e. Verifying a password is correct for a particular username).

dbConnect.java acts as a "middleman" between the Servlets and the database. By keeping all direct communications with the database (through SQL commands) were coded in the module dbConnect.java, the chance of errors were decreased and changes were made to the Servlets was reduced.

Appendix N

Milestones - Original

List each major milestone in chronological order. Assign **ONE** key team member that has ultimate responsibility to each milestone. Use up to two pages if necessary.				
	Description	Assigned to	Start Date	End Date
1.	Clear definition of project objectives, methodologies and software and hardware components that will be used: Collection and research of resources and papers that will be used for project. Obtain computer resources from Professor Zemel and obtain access priviledges to Artificial Intelligence labs	Andrew	01/09/2003	10/17/2003
2.	Implement Active Collaborative Filtering Methodology in Java. The back-end of the system will be coded in Java. This includes the calculations of EVOI for queries, model fitting for the probability model, and computing rating predictions. Its main purpose is to return a set of recommendations to a user given a	Cavan	10/17/2003	11/21/2003
3.	Database Implementation in MySQL: MySQL database will be created to store music information. Decisions on the music information to be displayed, types and categories of music to be used, and music sample format and size will be made based on research of related sites and considerations such as transfer speed	Andrew	10/17/2003	11/07/2003
4.	Webserver / GUI Setup using Jakarta Tomcat, Java Server Pages, Apache Server. A user friendly and informative interface will be designed and implemented. The GUI will provide features such as information about the music to be rated (mp3 sound clips, artist information etc.).	Bernard	10/17/2003	11/14/2003
5.	XML structure definition for music, users and ratings: A Document Type Definition structure will be created for use in the transfer of music information between the web server, application and database modules, and ultimately to the user. This step requires an understanding and coordination with the	Andrew	10/31/2003	11/17/2003
6.	Integration of web server and database: Interface for modular database component will have a focus on easy and fast access with webserver. Testing of speed of data retrieval will be looked at and optimizations and modifications to both components will be considered if necessary.	Andrew	11/14/2003	12/19/2003

7.	Integration of web server and application. In order to maximize the effectiveness of the system, the Web Server must interact quickly will the application. The speed of use will be tested, and modifications to both components will be made if necessary.	Bernard	11/21/2003	12/19/2003

3. Milestones (cont'd)

List each major milestone in chronological order. Assign **ONE** key team member that has ultimate responsibility to each milestone. Use up to two pages if necessary.				
	Description	Assigned to	Start Date	End Date
8.	Integration of application and database. (2 weeks) Enabling the Java code to successfully retrieve and update data to the database. Ensure that communication between the two components is smooth.	Cavan	01/04/2004	01/25/2004
9.	Testing, comparison and implementation of alternative recommender algorithms. (2 weeks) Comparison of "active" versus "non-active" approaches to collaborative filtering. Analyze their performances and accuracy to real life usage.	Cavan	01/26/2004	02/12/2004
10.	Oral Presentation: Summarization of progress will be condensed into presentation format. Visuals and aids will be considered and created to provide an effective emphasis and attract audience. Rehearsal of presentation and analysis of strengths and weaknesses	Andrew	02/13/2004	02/27/2004
11.	Design Fair Poster Presentation The poster will attempt to give an overall view of our design project while keeping in mind the audience will mostly be comprised of 3rd year ECE students. The results of our research will be displayed through the use of charts and graphs. A computer will also be available to provide a hands	Bernard	02/27/2004	03/16/2004
12.	Completion of group final report Intregration of the documentation of all components which make up the design project including diagrams, figures, references.	Cavan	03/16/2004	04/08/2004

Appendix O

Milestones - Updated

List each major milestone in chronological order. Assign **ONE** key team member that has ultimate responsibility to each milestone. Use up to two pages if necessary.				
	Description	Assigned to	Start Date	End Date
1.	Clear definition of project objectives, methodologies and software and hardware components that will be used: Collection and research of resources and papers that will be used for project. Obtain computer resources from Professor Zemel and obtain access privileges to Artificial Intelligence labs.	Andrew	09/01/2003	10/17/2003
2.	Implement Active Collaborative Filtering Methodology in Java. The back-end of the system will be coded in Java. This includes the calculations of EVOI for queries, model fitting for the probability model, and computing rating predictions. Its main purpose is to return a set of recommendations to a user given a database of user ratings.	Cavan	10/17/2003	01/21/2003
3.	Database Implementation in MySQL: MySQL database will be created to store music information. Decisions on the music information to be displayed, types and categories of music to be used, and music sample format and size will be made based on research of related sites and considerations such as transfer speed.	Andrew	10/17/2003	11/07/2003
4.	Web server / GUI Setup using Apache/Jakarta Tomcat Server and Java Servlets A user friendly and informative interface will be designed and implemented. The GUI will provide features such as information about the music to be rated (mp3 sound clips, artist information etc.).	Bernard	10/17/2003	12/13/2003
5.	Creation of Website to gather User Ratings using PHP, MySQL and HTML Website created and tested to gather approximately 75 user ratings and other information such as age, sex and mood for songs in the database. Information will be gathered and stored in multiple MySQL database tables to be used by the recommender system	Andrew	11/15/2003	01/17/2003
6.	Integration of web server and database. Interface for modular database component will have a focus on easy access by the web server. Extensive testing must be done to demonstrate the correctness of data stored/retrieved to/from the database. Modifications to the modules will be made if necessary.	Bernard	12/14/2003	01/24/2004

7.	Integration of web server and application In order to maximize the effectiveness of the system, the Web Server must interact quickly will the application. Integration will focus on the speed of communication between the modules. Extensive testing must be done to demonstrate the correctness of the data exchanged. Modifications to the modules will be made if necessary.	Bernard	12/14/2003	01/24/2004
8.	General Testing of web server / application / database functionality: As integration of web server, application and database progresses, general tests of the system will be conducted to determine boundary exceptions, broken links, correctness of algorithm implementation, and ease of communication and speed between modules.	Andrew	01/17/2004	02/13/2004
9.	Integration and testing of application and database. Enabling the Java code to successfully retrieve and update data to the database. Ensure that communication between the two components is smooth.	Cavan	01/04/2004	02/13/2004
10.	Coordination of Research Schedule. Decision on two methods of testing will be decided, numbers of test subjects to be used, and types of measuring scales to be used to determine accuracy and usefulness of recommendations.	Andrew	01/30/2004	02/13/2004
11.	Research, comparison and implementation of alternative recommender algorithms. Comparison of "active" versus "non-active" approaches to collaborative filtering. Analyze their performances and accuracy to real life usage.	Cavan	02/13/2004	03/05/2004
12.	Oral Presentation: Summarization of progress will be condensed into presentation format. Visuals and aids will be considered and created to provide an effective emphasis and attract audience. Rehearsal of presentation and analysis of strengths and weaknesses.	Andrew	02/13/2004	03/26/2004
13.	Design Fair Poster Presentation The poster will attempt to give an overall view of our design project while keeping in mind the audience will mostly be comprised of 3rd year ECE students. The results of our research will be displayed through the use of charts and graphs. A computer will also be available to provide a hands-on demonstration of our system.	Bernard	02/27/2004	03/16/2004
14.	Completion of group final report Integration of the documentation of all components which make up the design project including diagrams, figures, references.	Cavan	03/16/2004	04/08/2004