ABSTRACT

The application developed in this project acts as a personal assistant to an interested book user by helping him make the right choice in selecting the right book. The application observes and captures the user’s interest topic and preferences each time he uses the system. It learns to notify the user of any new pertinent changes taking place in real time while he is using the system. The autonomous tool developed in this project thus helps book customers transform their passive search into intelligent decisions while choosing books. The tool downloads a list of book records based on keyword and title searches and it displays scores and ratings for each book. Accordingly, the user can decide which book would be more useful to him among the rest in the database. The tool employs Java programming language along with MS-SQL database. Good usability was also taken into consideration while developing the application and suitable interface testing has been carried out to prove that.
# TABLE OF CONTENTS

Abstract.............................................................................................................................ii

Table of Contents.............................................................................................................iii

List of Figures..................................................................................................................vi

List of Tables....................................................................................................................vii

1. Introduction and Background......................................................................................1
   1.1 Introduction.............................................................................................................1
   1.2 Search Engines and Active Agents.................................................................1
   1.3 Classification of Agents.......................................................................................3
   1.4 Need for the Tool.................................................................................................3
   1.5 Java based Agent Environments.....................................................................4
   1.6 Summary of other Titles.......................................................................................4

2. Intelligent Tool for Consumer Decision Support.......................................................6
   2.1 External Aspects of Project...............................................................................6
   2.2 User Interfaces....................................................................................................6
      2.2.1 Login Interface...............................................................................................6
      2.2.2 New User interface.......................................................................................7
      2.2.3 Result Screen...............................................................................................8
      2.2.4 Display Records Screen...............................................................................8
      2.2.5 User Keyword Screen...............................................................................9
      2.2.6 Help Screen.................................................................................................9
      2.2.7 Main Screen...............................................................................................10
2.2.8 Delete Screen ................................................................. 10

3. System Design ................................................................. 11
  3.1 Main Components and System Requirements ....................... 11
    3.1.1 MS-SQL .................................................................. 11
    3.1.2 JAVA ..................................................................... 11
    3.1.3 System Architecture ................................................ 12
  3.2 System Design ............................................................... 13
    3.2.1 Flowchart .................................................................. 13
    3.2.2 Structure Chart ........................................................ 15
    3.2.3 Data Flow Diagrams .................................................. 16
      3.2.3.1 Level 0 Data Flow Diagram .................................... 16
      3.2.3.2 Level 1 DFD ........................................................ 16
  3.3 Data Base Tables ............................................................. 19
  3.4 Entity Relationship Diagram (ERD) .................................. 21
  3.5 Implementation ............................................................. 21
    3.5.1 Login Module ........................................................... 22
    3.5.2 Search String module ................................................ 22
    3.5.3 Find Record Module .................................................. 22
    3.5.4 Last Update module ................................................... 23
    3.5.5 Capture User Preferences Module ................................. 23
    3.5.6 Delete module .......................................................... 23

4. Evaluation and Results ......................................................... 24
  4.1 Usability Testing and Interface Evaluations ........................ 24
4.2 Report on Results of Interface Testing........................................25
  4.2.1 Questionnaire Survey.................................................................25
  4.2.2 Responses..................................................................................26
  4.2.3 Tasks.........................................................................................28

4.3 Software Testing.............................................................................30

5. Future Work.....................................................................................31

6. Conclusion.......................................................................................32

Bibliography and References.............................................................33

Appendix A Code Snippets.................................................................34

Appendix B Test Results.....................................................................40
LIST OF FIGURES

Figure 1.1 Different types of agents.........................................................3
Figure 2.1 Login Interface.................................................................7
Figure 2.2 New User Interface..........................................................7
Figure 2.3 Result Screen.................................................................8
Figure 2.4 Display Records Screen....................................................8
Figure 2.5 User Keyword Screen......................................................9
Figure 2.6 Help Screen.................................................................9
Figure 2.7 Main Screen..............................................................10
Figure 2.8 Delete Record Screen.......................................................10
Figure 3.1 Two Tier Architecture....................................................13
Figure 3.2 Flowchart for Design Phase.............................................14
Figure 3.3 Structure Chart for Design Phase....................................15
Figure 3.4 Context Level Diagram of the System............................16
Figure 3.5 Data Flow Diagram for Login Process.............................17
Figure 3.6 Data Flow Diagram for AddRecord Process.....................17
Figure 3.7 Data Flow Diagram for UpdateRecord Process................18
Figure 3.8 Data Flow Diagram for SearchFor Process.......................18
Figure 3.9 Data Flow Diagram for FindRecord Process.....................19
Figure 3.10 Data Flow Diagram for LastRecord Process....................19
Figure 3.11 ERD Model.............................................................21
LIST OF TABLES

Table 4.1 Tabular Results of Responses.................................................................29
Table 4.2 Tasks and Time taken.............................................................................32
Table 4.3 Summary of Results of Survey...............................................................33
1. INTRODUCTION AND BACKGROUND

1.1 INTRODUCTION

The project consists of the design and implementation of a personal assistant to help the book consumer in getting useful filtered information. The database contains a list of book records with their titles and descriptions and those can be downloaded into the application when required for the user to view them. The user can search for his interest topic in these records based on any number of string searches or title searches. Book records would then be displayed with cumulative score and ratings in a sorted manner that would be useful to his needs and requirements.

The main aspect of the smart agent implemented in this project lies in the fact that it adapts its behaviors to the changing environment, both in terms of the user’s objectives and the resources available to the agent. For example, it captures each user’s past search strings and titles and then reminds him of these past preferences and choices every time he logs in. The agent is always active in the sense that when a user logs in, it keeps track and informs him of any new records being added in the database by any other user at the same time. Furthermore, it informs the user of any new records being added that match his own interest since his past login. Above all, the tool adopts the principles of good human-computer interaction.

1.2 SEARCH ENGINES AND ACTIVE AGENTS

Existing search and retrieval engines provide only limited assistance to users in locating the relevant information that they need. The engines mainly provide passive data based on user search criteria. The amount of unnecessary information thrown to a user by
a search engine has an adverse effect on time and productivity. From an information
overload perspective, users can encounter so much information covering such a wide area
that they often abandon most of it, ending up with a limited set of retrieved documents
[Chen 1999]. The result is that users do not get what they are looking for. This is where
smart agents come into action.

Agents, like the one implemented in this project, transform passive search and
retrieval engines into active, personal assistants helping consumers make the right
decisions. There are several definitions of agents based on their task, autonomy, and
communication capabilities. Some of the major definitions and descriptions of agents are
included below.

Agent is an entity that perceives its environment through sensors and acts upon it
through its effectors. [Russell 2002]. Agents employ artificial intelligence techniques to
assist users with daily computer tasks, such as reading electronic mail, maintaining a
calendar, and filing information. Agents learn through example-based reasoning and thus
are able to improve their performance over time [Lashkari 1997]. Agents are
computational systems that inhabit a complex, dynamic environment. They sense and act
autonomously in this environment. By doing so, they realize a set of goals or tasks
[Russell 2002]. Agents are software robots. They can think and act on behalf of a user to
carry out tasks [Kuokka 1997].

The definition and description of an agent for this project is: agents are software
programs that are capable of interpreting user preferences, and choosing actions
autonomously that help the users in achieving their intended goals.
1.3 CLASSIFICATION OF AGENTS

The most natural way of classifying agents is based on the function they perform. User based interface agents try to act on what the users mean rather than what the users say [Ferreira 2005]. Information agents always supply information of interest to the user. However, they can quickly transform into a monster of information overload [Watson 1997]. Search agents look for documents on the Web while filter agents process incoming mail and other news postings. Domain specific assistants book a business trip, schedule a meeting or verify that a design does not violate any constraints [Bigus 2001].

Interface agents, as used in this project, act as personal assistants to help a user accomplish his tasks. They employ learning to adapt themselves to the preferences of the user. Figure 1.1 gives an illustrated overview of the different types of agents.

![Figure 1.1 Different Types of Agents](image)

1.4 NEED FOR THE TOOL

As stated above, smart agents filter information. This is necessary to avoid data overload for the online user. The agent implemented in this project accomplishes this task for a user interested in searching for a useful list of books. For example, if a user does a
“google” search of “java books”, he gets thousands of java book resources. There has to be a way for him to find out which of these books is more pertinent to his interest and preferences. This set of requirements is implemented in the project as a group of keywords which change as the situation demands. Once a new set of keywords is created, the agent remembers them by creating a new profile for the user. Hence every time, the user logs in, he can see his past searches in terms of keywords and titles. Based on these keyword matches, the agent comes up with values of score and rating for each searched book record in a sorted order. This approach helps the book user make the right decision.

1.5 JAVA BASED AGENT ENVIRONMENTS

The IBM Agent building and learning environment(ABLE) is a Java based framework for developing and deploying hybrid intelligent agents [Bigus 2001]. It provides a set of reusable JavaBean components along with flexible interconnection methods for combining those components to create software agents [Bigus 2001]. Gossip is a demonstration application of Tryllian, a mobile agent software. It uses learning technology to profile users’ preferences and to perform automated actions on behalf of the users [Patel 2001]. Voyager, from ObjectSpace is an agent enhanced Object Request Broker (ORB) that is written in Java [Bigus 2001]. These agents have mobility and autonomy provided in the base class.

1.6 SUMMARY OF REPORT

This report is divided into the following sections as below:

- The second section titled with the title of the project deals with the external aspects of the application in terms of user interfaces and project results.
• The third section titled “System Design” deals with the detailed description of the internal aspects of the project and the major components and their method of design and implementation.

• The fourth section “Evaluation and Results” deals with interface testing and also identifies portions where the report differs from the proposal.

• The fifth section “Future Work” deals with scope of future work related to this project

• The final section “Conclusion” summarizes the outcomes of the project
2. INTELLIGENT TOOL FOR CONSUMER DECISION SUPPORT

2.1 EXTERNAL ASPECTS OF PROJECT

The application is a multi user system with the option of running multiple versions of the application simultaneously. There is a login screen which verifies username and password. There is also a form for creating new users with all pertinent information. When the verified user reaches the main screen, the application helps him with various tasks. One such main task is that its shows all records matching the first time user’s interest topic as mentioned by him in registration form. Then, if the user would be logging in for the second or third time, he would be able to see all records that were recently added, by any user, which match his previous search criteria.

The agent would always be active, looking for record additions by any other user using the system at the same time. If it would find new records, then it would immediately inform the current user. The agent, thus basically performs like a personal assistant and helps the book consumers in making right decisions. Again, records are usually displayed in a sorted manner as the number of matches and percent match with the searched keywords. The percent match is calculated on the basis of the total number of words in each record title and description. This gives a rating value to each record suggesting its usefulness.

2.2 USER INTERFACES

2.2.1 Login interface

This interface verifies the username and password and lets him enter the main screen. On non-verification, an error message gets displayed. Figure 2.1 displays the screen.
Figure 2.1 Login Interface

2.2.2 New User Interface

This screen allows for the creation of a new user whose information gets added in database. All fields are mandatory and password verification is a must. Figure 2.2 displays the screen.

Figure 2.2 New User Interface
2.2.3 Result Screen

This screen comes up automatically displaying all new records added by any user that match a user’s previous search keywords. Here, based on what a user’s previous search keywords are, those new records are displayed which were added by any user since the first user’s last login time. Figure 2.3 displays the screen.

<table>
<thead>
<tr>
<th>Enter keyword(s) to search and click on ‘Search for String’</th>
<th>Previous history is also displayed</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>2 records added or updated since last login matching the keyword(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>war novel, romance, novel, curry</td>
</tr>
<tr>
<td>Title=A novel of birds</td>
</tr>
<tr>
<td>Description=A novel</td>
</tr>
<tr>
<td>LastUpdatedAt=2006-03-28 10:51:16.0</td>
</tr>
<tr>
<td>LastUpdatedBy=kalyan</td>
</tr>
<tr>
<td>Title=Frogs</td>
</tr>
<tr>
<td>Description=A novel</td>
</tr>
<tr>
<td>LastUpdatedAt=2006-03-28 10:54:36.0</td>
</tr>
<tr>
<td>LastUpdatedBy=kalyan</td>
</tr>
</tbody>
</table>

Figure 2.3 Result Screen

2.2.4 Display Records Screen

Here all records in database are displayed with their title and description. Figure 2.4 displays the screen.

| Key=Favorite Curry                                           |
| Description=Mothers recipee books                            |
|                                                               |
| Key=Programming Languages                                    |
| Description=A book describing aspects of different languages in a subtle way |
|                                                               |
| Key=India ABCs                                               |
| Description=A Book About the People And Places of India      |
|                                                               |
| Key=PHP                                                      |
| Description=A simple book                                    |
|                                                               |
| Key=Abnormal Psychology                                      |
| Description=This book introduce readers to abnormal psychology addressing the causes of psychopathology as well as issues in the t |
|                                                               |
| Key=Wave of Destruction                                      |
| Description=The Stories of Four Families and History's Deadliest Tsunami |

Figure 2.4 Display Records Screen
2.2.5 User Keyword Screen

In this screen records are displayed based on the number of keyword matches with the total number of words in the title and description in terms of a percentage. These results are produced in descending order. The Rating field signifies the usefulness of a book to the user. Figure 2.5 displays the screen

| Key = Exodus | Description = A novel | Number of Matches = 1 | Percent Match = 33.33 | Rating = Very Useful |
| Key = A novel of birds | Description = A novel | Number of Matches = 2 | Percent Match = 33.33 | Rating = Very Useful |
| Key = Frogs | Description = A novel | Number of Matches = 1 | Percent Match = 33.33 | Rating = Very Useful |
| Key = Favorite Curry | Description = Mothers recipe books | Number of Matches = 1 | Percent Match = 20.0 | Rating = Useful |
| Key = Pygmalion | Description = A novel of self acclamation | Number of Matches = 1 | Percent Match = 16.66 | Rating = Useful |

Figure 2.5 User Keyword Screen

2.2.6 Help Screen

In this screen, instructions are displayed to help the user gain a better understanding of the application.

Help

- Click 'Display all specific Record' with values field blank to display all record(s) from the database.
- Click 'Display all specific Record' with title field populated to locate specific record from the database.
- Click 'Search For' to get number of matches and percentage of match of the keywords entered from the database records.
- 'Rating' Field signifies how useful a book is to the user in terms of his search criteria.
- Click 'Add' to insert a new record in the database.
- Click 'Update' to update a record.
- Click "Clear" to empty the textfields.

Figure 2.6 Help Screen
2.2.7 Main Screen

This interface reflects the main working screen of the application with title bar, menus, search fields and results screen.

![Figure 2.7 Main Screen](image)

2.2.8 Delete Screen

Here users can delete any record he wants after searching for its title.

![Figure 2.8 Delete Screen](image)
3. SYSTEM DESIGN

3.1 MAIN COMPONENTS AND SYSTEM REQUIREMENTS

The main components of this project are J2SE version 1.4/1.5 and MS-SQL database. The entire application has been built using the java programming language while all necessary data has been stored in MSSQL acting as the back end. The project, since it deploys Java class files can be run under any system with a java run time environment installed. If an older version of java is being used like 1.2, the compiler might produce some warnings.

3.1.1 MS-SQL

MSSQL is a database platform providing more secure, reliable storage for both relational and structured data, enabling a user to build and manage all kinds of data applications. It has sample databases which can be used as test databases for any front end applications. It supports Microsoft's version of Structured Query Language (SQL), the most common database language [Tecuci 1998].

3.1.2 JAVA

The architecture neutral aspect of Java makes it ideal for programming on the Internet [Sierra 2003]. It allows a user to receive software from a remote system and execute it on a local system. This is done irrespective of the underlying hardware or operating system. Further, the portable interpreted nature of Java impacts its performance. This bytecode portability is what enables Java to be transported across a network and executed on any target computer system [Tecuci 1998]. Also, because Java applets can be downloaded from any system, security mechanisms exist within the java Virtual Machine to protect against malicious applets. The Java runtime system verifies
the bytecodes as they are downloaded from the network to ensure that they are valid
bytecodes and checks that the code does not violate any of the inherent restrictions placed
on applets [Eckel 2005]. Some of the restrictions are: applets can communicate only with
originating host and cannot run a local executable program.

One reason that Java applications can be autonomous is because they run as
separate processes. Intelligent agents are also autonomous programs. As such they are
always waiting, ready to respond to a change in the environment [Eckel 2005].

Prolog and C++ are also used for building AI applications, but as can be seen
from the above mentioned reasons, Java is now the preferred language for AI
applications. In fact, efforts have been made to integrate Prolog and Java in AI
development, mainly using the object oriented features of Java and the queries in Prolog
[Bigus 2001].

3.1.3 SYSTEM ARCHITECTURE

This project consists of a two-tier architecture where the application talks
directly to a server, with no intervening server. The two tier architecture is intended to
improve usability by supporting a forms-based, user-friendly interface [Sadoski 2005].
In general, the user system interface client invokes services from the database
management server. In many two tier designs, most of the application portion of
processing is in the client environment. The database management server usually
provides the portion of the processing related to accessing data [Sadoski 2005]. Figure
3.1 depicts such a typical two tier architecture:
In this project too, the user interface directly queries the database management system for processing functions, mainly involving SQL statements.

3.2 DESIGN PHASE

The designer needs a mechanism for explicitly and formally representing information that constraint some element of the design model [Pressman 2005]. This can be accomplished by structure diagrams and graphical figures. This project employs flow chart, structure chart and data flow diagrams for the design phase.

3.2.1 Flowchart

Figure 3.2 depicts the flowchart used. It first shows the login phase for a user where a new user can be created and identity verification is done. If it is a first time user, then he would see all records that match his interest topic. If not, then he would see any records added since his last login based on his previous search criteria. Based on his search criteria, he would see the records with their percent match and rating which would help him determine the usefulness of a book.
Figure 3.2 FlowChart for the Design Phase
3.2.2 Structure Chart

The second method employed is a structure chart. A structure chart depicts each software component in a box. It basically decomposes a problem and unlike a flowchart, does not have to be in any sequential order. Figure 3.3 depicts the chart.

Figure 3.3 Structure Chart for the Design Phase
3.2.3 Data Flow Diagrams (DFD)

DFDs show the flow of data from external entities into the system and how data moves from one process to another [Pressman 2005]. The design and functionality of the system is depicted in the DFDs below.

3.2.3.1 Level 0 Data Flow Diagram

Level 0 DFD or context level diagram depicts the overall data flow in the system as a whole. Figure 3.4 shows the flow of data in the system at a higher level.

![Figure 3.4 Context Level Diagram of the System](image)

3.2.3.2 Level 1 Data Flow Diagram

Level One Data Flow Diagram (DFD) explains the flow of data in the system in detail, compared to the context level DFD. The major processes of the system are shown in DFD fragments with inputs and outputs, to permit better descriptions of the processes. The following processes are the major components of the application.

**Login**

The *login screen* provides access to the application for any valid user. This screen accepts the username and password. These are passed to *login process* as an input. Here the username and password are validated against the *users* table from the database. Once the user is authenticated, the user is directed to the main screen of the application. If the user validation fails, the user is redirected to the *login Screen*. Figure 3.5 shows the Data Flow Diagram for this screen.
Figure 3.5 Data Flow Diagram for Login Screen

AddRecord

In this screen, all record data like title, description, cost, author are passed to AddRecord process and stored in the books table in database. Figure 3.6 shows the Data Flow Diagram for this screen.

Figure 3.6 Data Flow Diagram for Add Record Screen

UpdateRecord

In this screen, all record data like title, description, cost, author, ISBN are passed to UpdateRecord process if they need some modifications and stored in the books table in database. Figure 3.7 shows the Data Flow Diagram for this screen.
SearchFor
In this case, SearchFor process downloads all records from the books table which match user entered keywords. The Record module does the actual matching of keywords. The results are displayed in a descending order of matching keywords and their percent match in each record. A rating field signifies usefulness of each record. Figure 3.8 shows the Data Flow Diagram for this process.

FindRecord
This process downloads all records in database or specific records based on the title entered by user. Figure 3.9 shows the Data Flow Diagram for this process.
Figure 3.9 Data Flow Diagram for FindRecord Process

*Record_after_last_login*

This process displays all new records added in the database after the last login of the user and those that match his search criteria. The *getlastdate* process captures the last login of the user while *lastupdate* process captures the time the last record was added. *last_searchedwords* store the keywords. All these pass their information to the *LastRecord* module. Figure 3.10 shows the Data Flow Diagram for this process.

Figure 3.10 Data Flow Diagram for LastRecord Process

3.3 Data Base Tables

The database for this tool is MS-SQL. The database contains tables that store the records data and the users’ information who use this system. This subsection gives a brief description of the tables and fields in the database and their relationships. All the tables are normalized to avoid redundancy. The following is the list of tables in the database:

- Books
- Users
- User_log

The following is a brief description of the above tables:
Books:
This table has the following structure:

```
(title varchar(255),
  description varchar(1024),
  last_update datetime,
  last_updated_by varchar(50))
```

The title refers to the title of the book record, description details about the contents of the book and the time field is used to capture the time of addition of each record. This is necessary for the functionality of the application. last_updated field takes the value of that user who has added the most recent record.

Users:
This table has the following structure:

```
(username varchar(50),
  password varchar(50),
  first_name varchar(50),
  last_name varchar(50),
  address varchar(100),
  ph varchar(100),
  email varchar(100),
  gender varchar(10),
  interests varchar(100),
  last_login datetime)
```

Most of the fields relate to information about the user. The username/password field is checked for verification at all times. The time field is used to capture the user’s last login date and time. This is necessary for the functionality of the application.

User_log:
This table has the following structure:

```
(username varchar(50),
  last_search_words varchar(1024),
  last_search_title varchar(1024))
```
Here the user’s past preferences are captured in the form of his past searches of title/keywords and their date of searches. This is again necessary for the functionality of the application.

3.4 Entity Relationship Diagram (ERD)

ERDs are one of the main components of data model. Figure 3.11 depicts the ERD (Chen model) used in this system [Pressman 2005]:

![Figure 3.11 ERD Model](image)

3.5 Implementation

This section narrates the project implementation details. The entire project is divided into seven main modules. Each module is sub-divided into several units. The following is the list of modules:

- Login module
- Search String module
3.5.1 Login module

Here authentication is performed based on user id and password. Only if the information is valid, will the user be allowed to enter the main screen at this point. If authentication fails, an appropriate message is displayed. His last login date and time is also stored in this module. Appendix A contains the code snippet for this module.

3.5.2 Search String module

In this module, records are displayed on the basis of user search criteria like keywords and title. The user enters specific keywords and based on those words, titles are shown. The number of matches of keywords in the record and percent match of the number of keywords to the total number of words in the record are also displayed in descending order. Past preferences of user search and title are also captured. Appendix A contains the code snippet for this module.

3.5.3 Find Record/Price module

In this module, records are searched based on title field or in cases where no value is entered, all records are downloaded. The module also shows the price in amazon.com at that instant for the book. This helps the user in finding out the real time price of the book at any instant without having to browse the Internet and waste his precious time. The procedure has been accomplished by parsing and socket programming. Appendix A contains the code snippet for this module.
3.5.4 Last Update module

In this module, the last records added in database along with their date and time are tracked. This way the users would know if any records matching their past preferences (keywords) are being added or not. They would also know which user added them at what time. Thus the users would always be kept updated about new records being added. Appendix A contains the code snippet for this module.

3.5.5 Capture User Preferences module

In this module, all searched strings and title are tracked. The user has two options for searching: one is by title and the other is by a keyword. The user can search on the basis of his past keywords or he can enter new keywords and search as well. Each of his past keyword searches are stored along with the time of their search in this module. Appendix A contains the code snippet for this module.

3.5.6 Delete module

In this module, when a user enters a title, the remaining fields are displayed and the user can choose to delete that record permanently from the database. On successful deletion, an appropriate message comes up informing the user that the record has been deleted. If the connection fails, the user would be informed likewise. Appendix A contains the code snippet for this module.
4. EVALUATION AND RESULTS

For this project, both usability testing and software testing have been carried out. The testing activities are detailed below.

4.1 USABILITY TESTING AND INTERFACE EVALUATIONS

Usability testing is a major component of this project since it deals with user-agent communication in many ways. A user interface is well designed when the program behaves exactly how the user thought it would. The following factors play an important role in the designing of the dialogs and user interfaces [Dix 2003]:

1. Use simple and natural dialog
2. Speak the user’s language
3. Minimize the memory load
4. Be consistent
5. Provide feedback
6. Provide short cuts
7. Provide good error messages
8. Prevent errors

The testing of each module and component has been undertaken by not only the developer but also test users and students. These include library staff of the University who would want to use an application like the one developed in this project. The testing has been done through questionnaires. Also, none of the evaluators were a part of the development process, so before testing, they had no knowledge of the application. A report was prepared based on the findings. It has the set of questions given to each user
and their corresponding answers and a general description of the task or subtask and time taken to accomplish that task (average and planned).

4.2 REPORT ON RESULTS OF USABILITY TESTING

4.2.1 Questionnaire

The set of questions asked of each user in the survey are the following:

1. It is easy for me to enter my keyword searches and find corresponding books
   - Strongly agree   - Agree   - Neutral Disagree
   - Strongly Disagree

2. It is easy to read the results of keyword searches and determine the usefulness of a book
   - Strongly agree   - Agree   - Neutral Disagree
   - Strongly Disagree

3. It is easy to keep track of new records that are being constantly added which suit my preferences
   - Strongly agree   - Agree   - Neutral Disagree
   - Strongly Disagree

4. It is easy to use the interface without burdening myself with recall
   - Strongly agree   - Agree   - Neutral Disagree
   - Strongly Disagree

5. It is easy to see when I last used the system
   - Strongly agree   - Agree   - Neutral Disagree
   - Strongly Disagree

6. The system is implemented in a user friendly and helpful manner that suggests the application behaves as a personal assistant
   - Strongly agree   - Agree   - Neutral Disagree
   - Strongly Disagree

7. What additional features would you want to see in the application which you believe will make it work better as a personal assistant? Detail at least 2 features.

8. What did you like least about the application? Detail at least 2 features.

9. Would you want to use the same application for finding a book? Why or Why not?
4.2.2 Responses

There were seven evaluators to the questionnaire out of which five were from the library staff and two were students. The library staff uses similar applications and hence is in a good position to judge the application. There were five options to each question as noted in the previous section. *Strongly agree* carries a value of 10 points, *agree* has 5 points, *neutral* has 2 points while *disagree* and *strongly disagree* have -5 and -10 points respectively. The following Table 4.1 shows how each of the evaluators responded to each question and the weighted average for each question.

### Table 4.1 Tabular Results of Responses

<table>
<thead>
<tr>
<th>Questions</th>
<th>Responses(Points) and their Weighted Average</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Strongly agree(10)</td>
</tr>
<tr>
<td>1. It is easy for me to enter my keyword searches and find corresponding books</td>
<td>[5(10) + 1(5) + 1(2)] / 7 = <strong>8.1</strong></td>
</tr>
<tr>
<td>2. It is easy to read the results of keyword searches and determine the usefulness of a book</td>
<td>[4(10) + 3(5)] / 7 = <strong>7.8</strong></td>
</tr>
<tr>
<td>3. It is easy to keep track of new records that are being constantly added which suit my preferences</td>
<td>[5(10) + 2(5)] / 7 = <strong>7.8</strong></td>
</tr>
</tbody>
</table>
4. It is easy to use the interface without burdening myself with recall

\[ \frac{[2(10) + 5(5)]}{7} = 6.4 \]

5. It is easy to see when I last used the system

\[ \frac{[6(10) + 1(5)]}{7} = 9.1 \]

6. The system is implemented in a user friendly and helpful manner that suggests the application behaves as a personal assistant

\[ \frac{[4(10) + 2(5) + 1(2)]}{7} = 7.4 \]

The weighted average of each question has been calculated in the following manner: Respective points for a response have been multiplied by the number of responses for that question and then divided by the number of users. All the responses varied from strongly agree to agree and in two cases to neutral. This shows that the users were well satisfied with the interface of the application. Only for one question was the weighted average a little below the rest (6.4 for question no 4). All users except one agreed to use the system. The reason cited by that one user for not using the system is because she is already used to the library tool for book searching and hence would not like to switch to a new application. The one major feature that most users would like to see in the application is a connection to Web sites for comparison of prices of books or
for more information about a book. Among other minor changes they would like to see in the application is the use of buttons for searching instead of menus and the enabling of the “Enter” button for searching. All these responses have been included in Appendix B of the report. Overall, all users have liked the functionality and look of the application.

4.2.3 Tasks

The following tasks were asked to be performed by the evaluators:

Subtask 1 title/description
Search for a book with the title “Curry”
Download a book with the title you want.

Subtask 2 title/description
Search for books with the keywords “novel java”
Download a book with your choice of strings.

Subtask 3 title/description
Download all book records stored in database
The database has records stored with their title and description.

Subtask 4 title/description
Add a new title and description to the database
New records can be added at any time

Subtask 5 title/description
Search for all new books added by any user since your past login suiting your search preferences
New records added are displayed which match a user’s keywords

Subtask 6 title/description
Search for the number of matches and percent match of keywords in the record
The keyword matches are displayed as a percentage of the total no of words in record

Subtask 7 title/description
Search for the rating of a book
There is a rating of usefulness associated with each book

Subtask 8 title/description
Search for your last login time
The last login time of the user is displayed
The following Table 4.2 shows how the time taken by each task was noted. It shows the tasks and the average time to complete and the developer’s estimate of time (planned) to be used for each task by the evaluators.

<table>
<thead>
<tr>
<th>Scenario and Subtasks</th>
<th>Average Case</th>
<th>Planned</th>
</tr>
</thead>
<tbody>
<tr>
<td>SubTask 1</td>
<td>6 sec</td>
<td>5 sec</td>
</tr>
<tr>
<td>SubTask 2</td>
<td>6 sec</td>
<td>5 sec</td>
</tr>
<tr>
<td>SubTask 3</td>
<td>20 sec</td>
<td>5 sec</td>
</tr>
<tr>
<td>SubTask 4</td>
<td>15 sec</td>
<td>8 sec</td>
</tr>
<tr>
<td>SubTask 5</td>
<td>8 sec</td>
<td>4 sec</td>
</tr>
<tr>
<td>SubTask 6</td>
<td>8 sec</td>
<td>5 sec</td>
</tr>
<tr>
<td>SubTask 7</td>
<td>2 sec</td>
<td>2 sec</td>
</tr>
<tr>
<td>SubTask 8</td>
<td>3 sec</td>
<td>5 sec</td>
</tr>
<tr>
<td>SubTask 9</td>
<td>2 sec</td>
<td>3 sec</td>
</tr>
</tbody>
</table>

As can be seen from the above table, the time taken by the users is close to the planned time of the developer. For some tasks, they actually performed better than the planed time. Considering that the questionnaire responses were very positive and that the time taken for each task was reasonable, it can be concluded that the user interface was friendly and functional. The following Table 4.3 exemplifies this fact further.
Table 4.3 Summary of Results of Survey

<table>
<thead>
<tr>
<th>Positive Responses (Strongly Agree, Agree)</th>
<th>Negative Responses (Strongly Disagree, Disagree)</th>
<th>Neutral Responses</th>
<th>Total Responses</th>
<th>Percentage of positive responses</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>0</td>
<td>2</td>
<td>42</td>
<td>95%</td>
</tr>
</tbody>
</table>

4.3 SOFTWARE TESTING

Software testing involved the functional performance check. The software was tested to check if the tool performs all the functions of assisting a user in locating the book of his choice. The system was tested while it was being developed. The project was divided into modules and each module, in turn, was divided into many units. Several distinct test cases were conducted for each unit as well as for each module and finally for the entire tool. A coding standard was adopted and proper documentation of the code was followed for ease of maintenance. Once the units of a module were tested independently, the units were combined to form the module. Each module was tested individually to check if it achieved the objectives of the set of requirements. Once all the modules were developed, the modules were combined and tested.
5. FUTURE WORK

The application is currently a two-tier application where the client communicates directly with the server. This can be a disadvantage when many users would want to use the system or when one wants to access the application from any location. Hence, the future work that can be carried out for this project is to transform the architecture into a three-tier architecture. This way the application would not have to reside in individual machines, but instead can be uploaded to a Web server for anyone to access anytime from anywhere. Also, there would be no limitations on the number of users accessing the system.

Another function that could be implemented in the application is the addition of the feature of partial word searches. This way the user would not have to type in the whole title or keyword he is looking for. This would save time for the user and gives him more options.
6. CONCLUSION

The personal assistant application developed in this project captures user profile information and depending on user preferences, produces a set of results which match his search criteria and history. Search results as displayed as percent match of the entire record. The user can also keep track of new records being added at all times by any user, matching his search preferences. The application, thus, helps in locating the desired book for a user and giving him an idea of how useful the book would be to him. This prevents the user from browsing the Net and wasting his precious time to find his desired book. He can directly use this application for that purpose.

The application used DFD and flowcharts for design purposes and Java and MS-SQL for implementation purposes. Suitable interface testing was also conducted through questionnaires which show that the application has a friendly and functional interface. Overall, the application serves its purpose of helping the user find the right book.
BIBLIOGRAPHY AND REFERENCES


APPENDIX A CODE SNIPPETS

Login Module

private void login() {
    lStatus.setText("");
    String pass = new String(tPassword.getPassword());
    String lastLoginDate = getLastDate(tUsername.getText(), pass);
    if (lastLoginDate != null) {
        SQLServerManager sql = new SQLServerManager(fileName);
        sql.updateLastLogin(tUsername.getText());
        System.out.println("Last date: "+ lastLoginDate);
        gm = new GUIManager(fileName, tUsername.getText(), lastLoginDate);
        dispose();
    } else {
        lStatus.setText("Invalid Username/Password");
        tUsername.setText("");
        tPassword.setText("");
        tUsername.requestFocus();
    }
}

Search String Module

public void actionPerformed(ActionEvent e) {
    try{
        String stringVal = panel.getSearchStrData().getText().trim();
        if (!stringVal.equals("") ) {
            SQLServerManager sql = new SQLServerManager(fileName);
            String titleVal = panel.getTitleData().getText().trim();
            sql.saveLastQuery(username, titleVal, stringVal);
            Hashtable records = sql.getRecords();
            Enumeration enSql = records.keys();
            int hits = 0;
            while(enSql.hasMoreElements()){
                String key = (String)enSql.nextElement();
                Record r = (Record)records.get(key);
                String desc = r.getDescr();
                int i = r.numOfMatches(stringVal);
                double d = r.percentMatch(stringVal);
                if(i>0){
                    // Handle search results
                }
            }
        }
    } catch (Exception e) {
        // Handle exceptions
    }
}
Find Record/Price Module

public void actionPerformed(ActionEvent e){
  try {
    String title = panel.getTitleData().getText().trim();
    String string =
        panel.getSearchStrData().getText().trim();
    String descr = panel.getDescrData().getText().trim();
    SQLServerManager sql = new SQLServerManager(fileName);
    if (!title.equals("")) {
      sql.saveLastQuery(username,title,string);
      Hashtable records = sql.getRecords();
      Enumeration enSql = records.keys();
      int count = 0;
      while(enSql.hasMoreElements()){
        String key = (String)enSql.nextElement();
        Record r = (Record)records.get(key);
        String desc = r.getDescr();
        if(title.equalsIgnoreCase(key)){
          count++;
          String rating = "";
          if(d == 0)
            rating = "Useless";
          else if ((d>0) && (d<= 15))
            rating = "Slightly Useful";
          else if ((d>15) && (d<=25))
            rating = "Useful";
          else if (d>25)
            rating = "Very Useful";
          output.append("\t \t Rating = "
            +rating+"\n\n");
        }
      }
      if(hits == 0)
        output.append("\nNo records contain the entered search keywords");
    } else
      output.append("\nEnter keywords to search and click 'Search for String'\n");
  }catch(Exception ex){
    output.append(ex.toString());
  }
}
output.append("\nKey="+key+"\nDescription="+desc);
}
}
if(count == 0) output.append("No books found with
Title: "+title);
else{
    Hashtable records = sql.getRecords();
    Enumeration enSql = records.keys();
    while(enSql.hasMoreElements()){
        key = (String)enSql.nextElement();
        Record r = (Record)records.get(key);
        String desc = r.getDescr();
    }
}
if(writeAmazon)
    output.append("\nAmazon Price for this book = "+Amazon.init(title));
}
}
} catch (Exception ex) {
    output.append(ex.toString());
}

Last Update Module

public String updateLastLogin(String username){
    boolean ret = false;
    String retStr = "";
    Statement stmt;
    try{
        con= this.getConnection();
        if(con!=null){
            stmt = con.createStatement();
            Date date = new Date();
            String sdate = formatter.format(date);
            System.out.println("Date ->" + sdate + "<-"PRECEDING 36
        ret = stmt.executeUpdate("update users set last_login=sdate + " where username=username+'");
        retStr = "User record updated
        successfully!!!";
            closeConnection();
        }else
            retStr = "Error: No active Connection";
    }catch(Exception e){
        e.printStackTrace();
        retStr = "User record update did not
        succeed!!!";
    
    

36
Capture User Preferences Module

public String getLastString(String username, int lastN) {
    boolean ret=false;
    String lastTitle ="";
    ResultSet results = null;
    Statement stmt;
    int updateCount = 0;
    try{
        con= this.getConnection();
        if(con!=null){
            stmt = con.createStatement();

            ret = stmt.execute("select last_search_words from user_log where username='"+ username + "' order by timestamp desc");
            if (ret == true) {
                results = stmt.getResultSet();
            } else {
                updateCount = stmt.getUpdateCount();
                return "";
            }
            int counter = 0;
            while (results.next() &&
                    (counter<lastN)) {
                if (counter != 0) {
                    lastTitle += ",";
                }
                lastTitle +=
                results.getString("last_search_words") ;
                System.out.println("counter =" + counter + "
                lastTitle = " + lastTitle);
                counter++;
            }
            results.close();
            closeConnection();
        }else
            System.out.println("Error: No active Connection");
    }catch(Exception e){
        e.printStackTrace();
    }
    return lastTitle;
}
public String deleteRecord(DeleteRecordWindow window, String title, String descr, String author, String estimatecost, String ISBN, String topic, int flag) {
    boolean ret = false;
    String retStr = "";
    System.out.println(author);
    System.out.println(estimatecost);
    System.out.println(ISBN);
    System.out.println(topic);
    Statement stmt;
    try {
        con = this.getConnection();
        if (con != null) {
            stmt = con.createStatement();
            Date date = new Date();
            String sdate = formatter.format(date);
            System.out.println("Date ->" + sdate + "<-");
            // My Change
            if (flag == 1) {
                ResultSet results = null;
                ret = stmt.execute("select * from books where title='" + title + "'");
                if (ret == true) {
                    results = stmt.getResultSet();
                    int i, rowcount = 0;
                    String estimatedcoststring = null, descriptionstring = null, authorstring = null, ISBNstring = null, topicstring = null;
                    while (results.next() && rowcount < 100) {
                        descriptionstring = results.getString("description");
                        authorstring = results.getString("author");
                        estimatedcoststring = results.getString("estimatecost");
                        topicstring = results.getString("topic");
                        // DeleteRecordWindow.Update.setEnabled(true);
                        window.Delete.setEnabled(true);
                    }
                }
                window.txtdescription.setText(descriptionstring);
            }
        }
    }
}
window.txtauthor.setText(authorstring);
window.txtestimatecost.setText(estimatedcoststring);
window.txtISBN.setText(ISBNstring);
window.txttopic.setText(topicstring);

/*
DeleteRecordWindow.txtdescription.setText(descriptionstring);
DeleteRecordWindow.txtauthor.setText(authorstring);
DeleteRecordWindow.txtestimatecost.setText(estimatedcoststring);
DeleteRecordWindow.txtISBN.setText(ISBNstring);
DeleteRecordWindow.txttopic.setText(topicstring);*/

if(flag == 2)
{
    //My Change End

    ret = stmt.execute("delete from books where title='"+title+"'");

    retStr = "Book record deleted successfully!!!";
}
else
    closeConnection();

}catch(Exception e){
e.printStackTrace();
    e.printStackTrace();
    retStr = "Book record deletion did not succeed!!!";
}
APPENDIX B TEST RESULTS

Test results have been included in this section.