NATURAL LANGUAGE SYSTEM FOR INFORMATION RETRIEVAL

PROJECT REPORT

Submitted in partial fulfillment of the requirements

for the award of the degree of

BACHELOR OF ENGINEERING IN

COMPUTER SCIENCE AND ENGINEERING

OF THE BHARATHIAR UNIVERSITY, COIMBATORE

Submitted by

S. PRADEEP (0027K0190)
P. RAMYA DEVI (0027K0197)

Under the Guidance of

Mr. S. MOHANAVEL , B.E.,M.B.A.,D.C.P.A.
SENIOR LECTURER

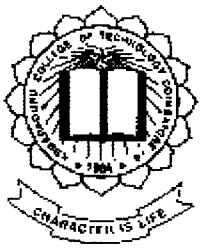
MARCH 2004

Department of Computer Science and Engineering

Kumaraguru College of Technology

COIMBATORE – 641 006

CERTIFICATE



KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE – 641 006.

Approved by AICTE, New Delhi - Accredited by NBA

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the project entitled

NATURAL LANGUAGE SYSTEM FOR INFORMATION RETRIEVAL

has been submitted by

S. PRADEEP

P.RAMYA DEVI

in partial fulfillment of the requirements for the award of the degree of Bachelor of Engineering in Computer Science and Engineering of the Bharathiar University, Coimbatore – 641 046 during the academic year 2003-2004

S. JL — 22/03/04

Head of the Department

Project Guide

Submitted for the university examination held on 23-03-2004

D. Chandra 24/3/'04

Internal Examiner

External Examiner

DECLARATION

DECLARATION

We hereby declare that the project entitled "NATURAL LANGUAGE SYSTEM FOR INFORMATION RETRIEVAL" submitted to the Bharathiar University,Coimbatore in partial fulfillment of the requirements for the award of the Degree of Bachelor of Engineering in Computer Science is a record of original work done by us under the supervision and guidance of Mr.S.Mohanavel,Senior Lecturer, Department of Computer Science and Engineering, Kumaraguru College of Technology, Coimbatore-641 006 and that it has not formed the basis for the award of any Degree / Diploma / Associate ship / Fellowship or other similar title to any candidate of any university.

Coordinator 23/3/'04

(Mrs.D.Chandrakala M.E.)

Candidate

(S.Pradeep

P.Ramya Devi)

Date: 23/3/2004

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

The exhilaration achieved upon the successful completion of any task should be definitely shared with the people behind the venture. This project is an amalgam of study and experience of many people without whose help it would not have taken shape.

At the onset, I take this opportunity to thank the management of our college for having provided us excellent facilities to work with. I express my deep gratitude to our Principal **Dr.K.K.Padmanabhan B.Sc (Engg), M.Tech.** for ushering us in the path of triumph.

We are always thankful to our beloved Professor and the Head of the Department, **Dr.S.Thangasamy B.E.(Hons).** whose consistent support and enthusiastic involvement helped us a great deal.

We are greatly indebted to my beloved guide **Mr.S.Mohanavel B.E.,M.B.A.** Senior Lecturer, Department of Computer Science and Engineering for his excellent guidance and timely support during the course of this project. As a token of my esteem and gratitude, I honor him for his assistance towards this cause.

We also thank our project coordinator **Mrs.D.Chandrakala M.E.** and our beloved class advisor **Mrs.M.S.Hema Guptha B.E.** for their invaluable assistance.

We also feel elated in manifesting my deep sense of gratitude to all the staff and lab technicians in the Department of Computer Science and Engineering.

We feel proud to pay my respectful thanks to my parents for their enthusiasm and encouragement and also I thank my friends who have associated themselves to bring out this project successfully.

CONTENTS

CONTENTS

SYNOPSIS

SYNOPSIS

The project entitled "**NATURAL LANGUAGE SYSTEM FOR INFO RMATION RETRIEVAL** is an intelligent interactive information retrieval system, which handles the user quires as natural language through voice as input and retrieves texts/documents accordingly.

Project is implemented for Library Book Management under natural language framework. It works on three concepts:

1. Speech Recognition
2. Semantic Analysis
3. Information Retrieval

This produces a keyboard free voice interactive environment, in which the queries are analyzed via voice processing. Input to the system is voice signal (user's natural language query) which is recognized to retrieve the accurate information efficiently from the database.

The targets are

(1) To make a user free from the burden of finding suitable keywords and their logical combination.

(2) To select the texts by making use of the information other than keywords, naturally contained in a natural language query.

(3) To automatically select specific topic of texts as per the query. The system solving two former targets consists of the models of a task domain, the parsers for query and text, the keyword generation expert and the content matcher.

INTRODUCTION

1. INTRODUCTION

The current age of information is characterized by the convergence of computing, communication, and content.

Round the clock, ubiquitous access to information and services is increasingly becoming a necessity in our daily lives. It is desirable to develop a **Human-Computer interface** which enables a broad range of users to consult computer for electronic information in a variety of application domains.

1.1 EXISTING SYSTEM

The existing system for "Library Management" involves manual processing in book handling. It holds demerits which includes adding books, updating, searching, etc. Even though many Computerized library management systems exist they are purely keyboard based and not an interactive and automatic.

The above demerits are rectified in this system. Here the input is given through voice and retrieves actual information.

1.2 PROPOSED SYSTEM AND ADVANTAGES

The Goal of the project is that, " The accurate capturing of the natural language query spoken by the user to retrieve data ". The captured signal are recognized and then translated to the digital signal that is produced as input to the system.

The project is implemented for "**Library Management System**". Based on the voice signal, the data are retrieved from the database which is more accurate.

The system provides keyboard free and Human-Computer Interface environment which is more comfort for managerial activities.

SOFTWARE REQUIREMENT AND ANALYSIS

2. SOFTWARE REQUIREMENT AND ANALYSIS

System study is an activity that encompasses most of the tasks that we have collectively called computer system engineering. System study is conducted with the following objectives.

- Identify the needs.
- Evaluate the system concept for feasibility.
- Perform economic and technical analysis.
- Allocate function to hardware, software, people and other system elements.
- Create a system definition that forms the foundation for all subsequent engineering works.

2.1 PRODUCT DEFINITION

The "**NATURAL LANGUAGE SYSTEM FOR INFORMATION RETRIEVAL**" an intelligent interactive information retrieval system, which handles the user quires as natural language through voice as input and retrieves texts/documents accordingly to user's natural language query.

2.2 CONCEPTS BEHIND

NATURAL LANGUAGE PROCESSING

The NLU (Natural Language Understanding) technology can be interfaced with speech which can handle user queries in spoken form. "Understanding" a natural language query refers to the computer's ability to transform the verbal form into machine readable semantics.

To avoid forcing searchers to memorize query languages, some systems allow them to type in a question, and use that as the query: this is known as "Natural Language Processing" (NLP). The simplest processing just removes stop words and uses a statistical approach. Some sophisticated systems try to extract concepts using linguistic analysis, and match those against concepts extracted by the indexer.

2.2.1 SPEECH RECOGNITION provides computers with the ability to listen to spoken language and to determine what has been said. In other words, it processes audio input containing speech by converting it to text. This is done with the help of IBM-Via Voice, which accepts user queries as input through voice.

This is processed and converted into text which is fed as input to IBM-JS (Java Speech Engine), which holds the speech grammar file (.gram file). This '.gram' file holds the predefined keywords.

2.2.2 SEMANTIC ANALYSIS is done by equating the text output from IBM-JS with the grammar (.gram) file. Thus the digital signal is lexically analyzed to produce tokens. Semantics of the tokens are identified and compared with the grammar (.gram) file is in javax.speech package *and* responds to the user needs by displaying the file.

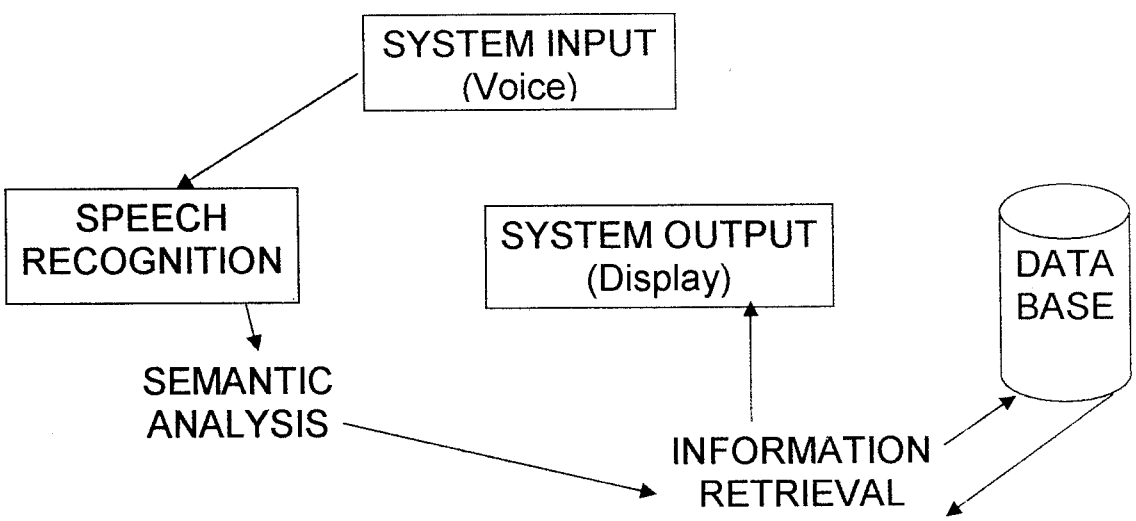
The keywords those matches the text are captured and their corresponding documents are retrieved from the database.

2.2.3 INFORMATION RETRIEVAL is the academic discipline which underlies computer-based text search tools. It tends to concentrate on mathematical models and algorithms for retrieval quality, but there is a great deal of valuable research in the field.

NLP can improve the ranking of documents because it has done a better job of matching the meaning and the intention of the query. It has more evidence of what is really relevant on which to base a relevance judgment.

The product provides a reliable, robust and efficient means of speech recognition. The project basically an interactive program which recognizes speech, transfers it to machine readable form (digital).

FUNCTIONAL DIAGRAM



2.3 PROJECT PLAN

- System Engineering ---- Sep `03
- Requirement Gathering & Analysis ---- Oct`03
- Designing ---- Nov`03
- Coding ---- Dec`03 & Jan`04
- Validation & Testing ---- Feb`04
- Documentation ---- up to Mar`04

SOFTWARE REQUIREMENT SPECIFICATION

3. SOFTWARE REQUIREMENT SPECIFICATION

3.1 PURPOSE

The primary purpose of the Software Requirements Specification (SRS) is to document the previously agreed to functionality, external interfaces, attributes, and the performance of the NATURAL LANGUAGE SYSTEM FOR INFORMATION RETRIEVAL. This specification is the primary document upon which all of the subsequent design source code, and test plan will be based.

3.2 SCOPE

The scope of this document, Software Requirement Specification (SRS) is to describe the requirements definition effort. The SRS documentation for Natural Language System for information retrieval describes the functions, external interfaces, attributes, and performance issues specified in the product definition.

3.3 PRODUCT OVERVIEW AND SUMMARY

The product provides a reliable, robust and efficient means of speech recognition. The project basically an interactive program which recognizes speech, transfers it to machine readable form (digital). Thus the digital signal is lexically analyzed to produce tokens. Semantics of the tokens are identified and compared with the grammar (.gram) file is in javax.speech package and responds to the user needs by displaying the file.

3.4 DEVELOPMENT AND OPERATING ENVIRONMENT

The development environment gives the minimum hardware and software requirements.

HARDWARE SPECIFICATION

Processor – Pentium III

RAM – 256 MB

Hard Disk – 10 GB

Floppy Drive – 1.44 MB

Monitor – 14" Monitor

SOFTWARE SPECIFICATION

Operating System – Windows 98, Linux.

Platform – Java

Software-IBM Via Voice, IBM Java Speech Engine

3.4.1 JAVA

Java is a technology that makes it easy to build distributed application over the network. Java allows the user to do the following:

- Writing robust and reliable programs.
- Build on application on almost any platform and run the application on any platform without recompiling the code.
- Distribute your application over the network in a secured fashion.

FEATURES

There are six features in Java that makes java as a power tool for applications,

- Security.
- The core API.
- Open Standard.
- Distributed and Dynamic.

- Object oriented.
- Multithreaded

3.4.2 SPEECH APPLICATION

The existing capabilities of the Java platform make it attractive for the development of a wide range of applications. With the addition of the Java Speech API, Java application developers can extend and complement existing user interfaces with speech input and output. For existing developers of speech applications, the Java platform now offers an attractive alternative with:

- **PORTABILITY:** the Java programming language, APIs and virtual machine are available for a wide variety of hardware platforms and operating systems and are supported by major web browsers.
- **POWERFUL AND COMPACT ENVIRONMENT:** the Java platform provides developers with powerful, object-oriented, garbage collected language which enables rapid development and improved reliability.
- **NETWORK AWARE AND SECURE:** from its inception, the Java platform has been network aware and has included robust security.

3.4.3 JAVA SPEECH API

The Java Speech API defines a standard, easy-to-use, cross-platform software interface to state-of-the-art speech technology. Two core speech technologies are supported through the Java Speech API: **speech recognition** and **speech synthesis.**

To use the Java Speech API, a user must have certain minimum software and hardware available. The individual requirements of speech synthesizers and speech recognizers can vary greatly.

- **SPEECH SOFTWARE:** A JSAPI-compliant speech recognizer or synthesizer is required.

- **SYSTEM REQUIREMENTS:** most desktop speech recognizers and some speech synthesizers require relatively powerful computers to run effectively.

- **AUDIO HARDWARE:** Speech synthesizers require audio output. Speech recognizers require audio input. Most dictation systems perform better with good quality sound cards.

- **MICROPHONE:** Desktop speech recognition systems get audio input through a microphone. Some recognizers, especially dictation systems, are sensitive to the microphone. Headset microphones usually provide best performance, especially in noisy environments. Table-top microphones can be used in some environments for some applications.

3.4.4 SPEECH RECOGNIZER

The major steps of a typical speech recognizer are:

- **GRAMMAR DESIGN:** recognition grammars define the words that may be spoken by a user and the patterns in which they may be spoken. A grammar must be created and activated for a recognizer to know what it should listen for in incoming audio.
- **SIGNAL PROCESSING:** analyze the frequency characteristics of the incoming audio.
- **PHONEME RECOGNITION:** A phoneme is a basic unit of sound in a language. It compares the spectrum patterns to the patterns of the phonemes of the language being recognized.
- **WORD RECOGNITION:** compare the sequence of likely phonemes against the words and patterns of words specified by the active grammars.

- **RESULT GENERATION:** provide the application with information about the words the recognizer has detected in the incoming audio. The result information is always provided once recognition of a single utterance is complete, but may also be provided during the recognition process. The result always indicates the recognizer's best guess of what a user said.

3.4.5 IBM VIA VOICE

IBM Via Voice is a speech recognition and synthesis software package developed by IBM. It receives the input from the microphone and can send the output to the speaker. When the computer is in a state to accept speech input, the system interprets what the user say. Via Voice will capture the words as streams of commands or dictated text. Using the Voice Centre in the Via Voice, the user can enroll and train the system by speaking about 476 sentences provided by it. After the voice registration is completed, the system knows how the user speaks out each word and provides accurate speech recognition. It is basically a speech environment provided, on which the Java Speech API works.

Speech for Java is a Java programming SDK for speech that gives Java applications access to the IBM Via Voice speech technology, The SDK supports speech to text recognition, dictation and text to speech synthesis based on the IBM Via Voice technology. The SDK is an implementation of the Java Speech API.

3.4.6 GRAMMAR

A **grammar** defines what a recognizer should listen for in incoming speech. Any grammar defines the set of tokens a user can say (a token is typically a single word) and the patterns in which those words are spoken.

The Java Speech API supports two types of grammars: **rule grammars** and **dictation grammars.** These grammars differ in how patterns of words are defined. They also differ in their programmatic use: a rule grammar is defined by

an application, whereas a dictation grammar is defined by a recognizer and is built into the recognizer.

A **rule grammar** is provided by an application to a recognizer to define a set of rules that indicates what a user may say. Rules are defined by tokens, by references to other rules and by logical combinations of tokens and rule references. Rule grammars can be defined to capture a wide range of spoken input from users by the progressive combination of simple grammars and rules. Rule grammars follow the style and conventions of grammars in the Java Speech Grammar Format. Any grammar defined in the JSGF can be converted to a Rule Grammar object. The Java Speech Grammar Format has been developed for use with recognizers that implement the Java Speech API.

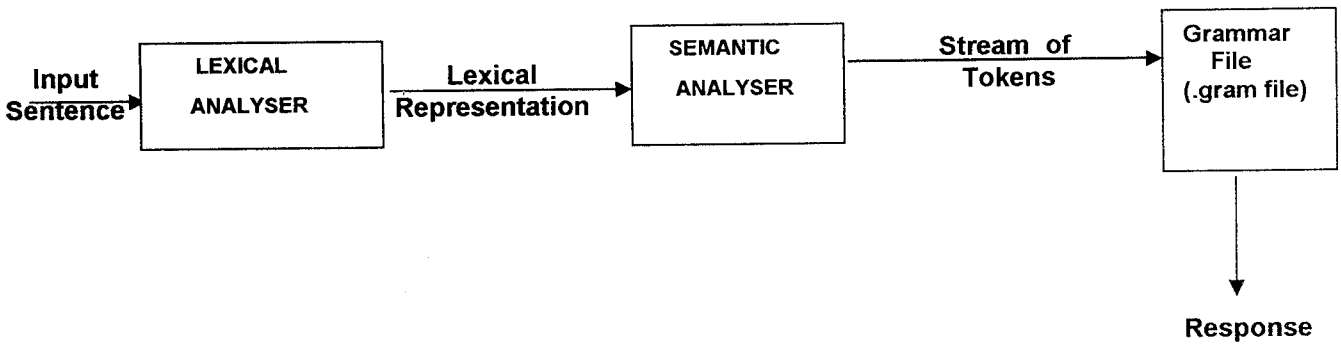
A **dictation grammar** is built into a recognizer. It defines a set of words (possibly tens of thousands of words), which may be spoken in a relatively unrestricted way. Dictation grammars are closest to the goal of unrestricted natural speech input to computers. Although dictation grammars are more flexible than rule grammars, recognition of rule grammars is typically faster and more accurate.

3.5 FUNCTIONAL SPECIFICATION

The speech input to the system is accepted by IBM-Via Voice. This is converted into text and this text is lexically analyzed into stream of tokens. This is done by the lexical analyzer in Via Voice.

The Syntactic analyzer combines the tokens to build a meaningful word. The newly built word is semantically analyzed by the JavaX package in IBM-JS (Java Speech Engine). The words are compared with the predefined keywords in the grammar file (.gram). the grammar file holds the tags in which the keyword are defined. This tags are called Embedded Grammar Tags (EGT).

SPEECH ACT ANALYSIS



The EGT(Embedded Grammar Tag) resides in the server that performs the semantic analysis. This tool gives a user the flexibility. The semantically analyzed tokens are combined to form an actual spoken text.

Information Retrieval – The semantically analyzed text is then compared with the database and it retrieves the most appropriate information

3.6 EXCEPTION HANDLING:

- The exception handling starts from the user validation in the Entry form where the User Id and the Password is checked.
- The validation is done for numerals and characters while adding and updating the books.
- The range of the names for each entry is validated.
- Validation for the wrong entry (unavailable books) is done in searching the books.
- Validation for adding empty form to the database.
- Redundancy of unique id is validated.

SYSTEM DESIGN

4. SYSTEM DESIGN

The system design phase is an interesting phase. It provides the way the information is to be fed and how the output is to be obtained. The design goes through logical and physical stages of development. Logical stage involves preparing the input and output preparation of the system. The physical stage details the hardware and the system implementations.

4.1 INPUT DESIGN

Reliable input design ensures accurate output from the system. The system to be designed is configured with the following data:

- User Name
- Password
- Speech input from the microphone

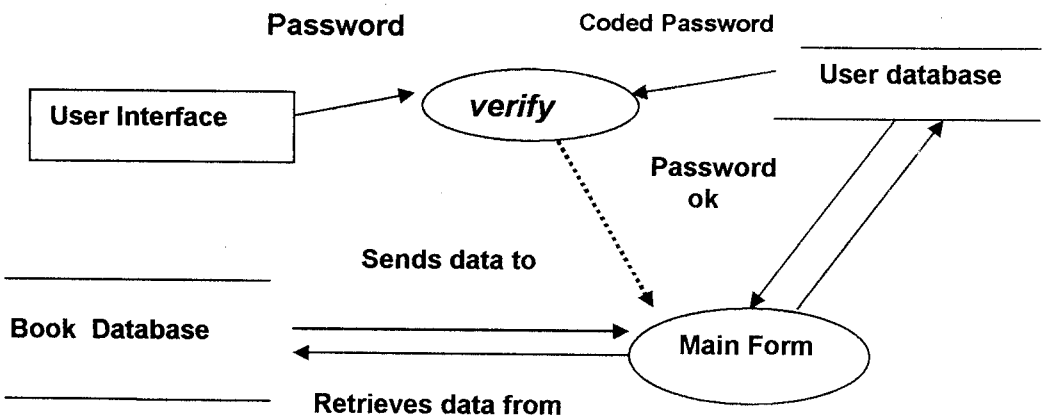
Care is taken that the inputs to the system are interactive and user friendly.

4.2 PROCEDURE DESIGN

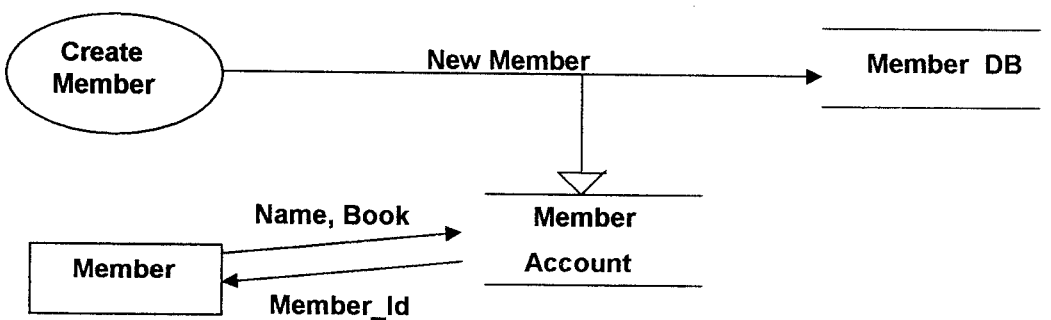
EXTERNAL INTERFACES AND DATAFLOW

The external interface includes frame where options to set encoding process and to receive password in Viewer part where as in server side a user input is obtained to process input events in DOS screen

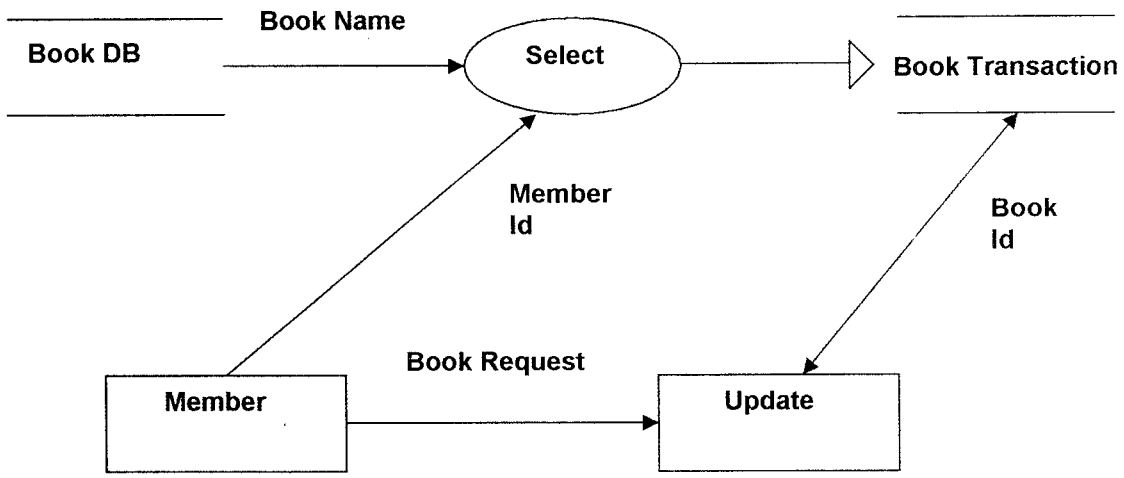
INTEGRATED PROCESS



CREATE ACCOUNT



RETRIEVAL PROCESS



4.3 OUTPUT DESIGN

Outputs from the system are required to communicate the result with the database for accurate retrieval . The outputs of the system are the following:

- Lexically analyzed tokens
- Machine readable semantics
- Information from the database
- File display

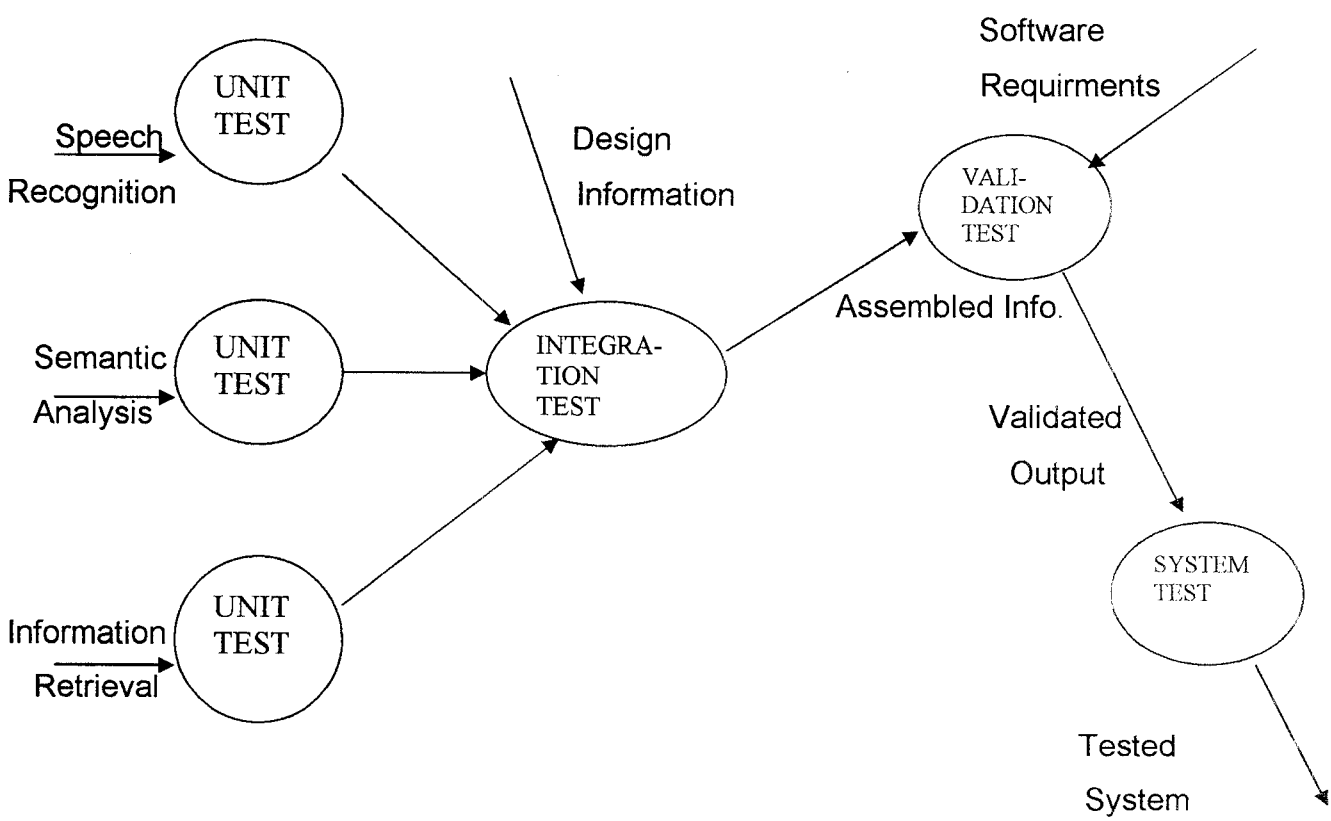
SYSTEM TESTING

5. SYSTEM TESTING

System Testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding.

The process of testing is done with intent of **finding an error** in the system or to be working according to specification and that performance requirements appear to have been met. Additionally the data collected for testing provides a good indication of software reliability and some indication of software quality.

TESTING STEPS



UNIT TESTING

The unit testing is done in order to focus even on the smallest unit of software design. In this system even the length of characters, naming conventions, password and the id of the user are tested.

In Recognition module unit testing is done by initially recording and retrieving all the keywords in via Voice software.

In Semantic analysis, testing is done by checking all the keywords in the grammar file.

In Retrieval ,testing is done by checking whether the data entered is matched with data type, etc.

INTEGRATION TESTING

This test is performed to test for inadvertent loss of data. This test uncovers errors associated with interfacing.

Here the interface between IBM-Via Voice and IBM-Js is checked. Testing is done between javax package (stream of tokens) and the grammar file(tags) is done. The data flow between the grammar file and the database is checked.

VALIDATION TESTING

The input areas are checked for validation existence say, Range and Reasonableness is tested.

CONDITION TESTING

The various conditions through with dataflow passes are tested. The Looping is done until the condition is made true. Here looping occurs until tokens are matched.

SYSTEM TESTING

The whole software is tested as one unit. The entire system is checked after integration of all the modules and the error that occurs at the execution time handled efficiently.

CONCLUSION

6. CONCLUSION

The complete design and development of the system "Natural Language System for Information Retrieval" is presented in this dissertation. The system has user-friendly features. It is possible for any user to use this system.

The programming techniques used in the design of the system provide a scope for further expansion and implementation of any changes, which may occur in the future. Finally, users can creatively expand the semantic reach of web age content simply by creating new EGT's that reflect potential queries.

This system is developed with the specifications and abiding by the existing rules and regulations of the company.

Since the requirements of any organization and their standards are changing day to day the system has been designed in such a way that its scope and boundaries could be expanded in future with little modification. As a further enhancement this system can be integrated with any other system.

This system has been developed using JAVA. The main aim behind the development of this system is to provide cross-platform, accurate speech recognition, and accurate information retrieval.

FUTURE ENHANCEMENT

7. FUTURE ENHANCEMENTS

The semiautomatic approach for grammar induction to achieve language understanding in restricted domains but it can be extended out of domain. It will be devoted towards improving the grammar quality and examining probability across languages and domain.

1. The same system can be implemented for any language.
2. The database entry can be entered through voice.
3. The System can be extended for Web Browsing activities.
4. Speech Synthesis can be adopted as a part of system.

REFERENCES

8. REFERENCES

1. Herbert Schildt, Java 2-Complete Reference, Tata McGraw Hill Edition, 2002.

2. Pergamon / Elsevier ,Information Processing & Management ,Science International Journal, US

3. Ricardo Baeza-Yates & Berthier , Ribero-Neto, Addison-Wesley Longman,Modern Information Retrieval ,International journal,1998

4. C.J.van Rijsbergen,Information Retrieval, 2nd Edition [textbook],1979

WEB SITES VISITED

1. www.search.cpan.org ---- 05/09/2003

2. karolina@linguistlist.org ---- 03/10/2003

3. www.Askjeeves.com ---- 08/01/2004

APPENDIX

APPENDIX

SAMPLE SOURCE CODE

```
// MAIN FORM
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.sql.*;
import javax.speech.*;
import javax.speech.recognition.*;
import javax.speech.synthesis.*;
import java.io.FileReader;
import java.util.Locale;
import java.util.Properties;
import java.util.Enumeration;
import java.util.StringTokenizer;

// TO LIST THE TITLE OF THE BOOKS
class Titlebooks extends Frame
{
        List l1;
        public void dbConnection(String text1)
        {
                try
                {
                        Driver drv=new sun.jdbc.odbc.JdbcOdbcDriver();
                        Connection
                    conn=DriverManager.getConnection("jdbc:odbc:netdb");
                        Statement stmt=conn.createStatement();
                        ResultSet rs=stmt.executeQuery("select * from Books where
                    title="+text1);
```

```
                boolean flag1=false;
                boolean flag2=false;
                while(rs.next())
                {
                        u=rs.getString(3);
                        l1.add(u);
                }
        }
        catch(SQLException sq)
        {
                System.out.println(sq.getMessage());
        }
}

public Titlebooks(String hi)
{
        setTitle(hi);
        l1=new List();
        setSize(400,400);
        show();
        l1.setBounds(100,100,200,300);
        add(l1);
        dbConnection(hi);
        Button b1=new Button();
        add(b1);
        remove(b1);
        addWindowListener(new WindowAdapter()
        {
                public void windowClosing(WindowEvent we)
                {
                        hide();
                }
        });
}
}
```

```
// SPEECH TO TEXT CONVERSION
class SpeechToText extends ResultAdapter
{
        static Recognizer rec;
        static String sp;

        public void Begin(String gramfile)
        {
                try
                {
                        rec=Central.createRecognizer(new
                      EngineModeDesc(Locale.ENGLISH));
                        rec.allocate();
                        FileReader reader = new FileReader(gramfile);
                        RuleGrammar gram = rec.loadJSGF(reader);
                        gram.setEnabled(true);
                }
                catch (Exception e)
                {
                        e.printStackTrace();
                }
         }
         public void resultAccepted(ResultEvent e)
         {
                sp="";
                Result r = (Result)(e.getSource());
                ResultToken tokens[] = r.getBestTokens();

                for (int i = 0; i < tokens.length; i++)
                        sp=sp+tokens[i].getSpokenText()+" ";

       System.out.println("Currently spoken word "+sp);
       }
          public void Kill()
          {
```

```
            System.exit(0);
        }
        public String MainMethod()
        {
            try
            {       sp="";
                    rec.addResultListener(new SpeechToText());
                    rec.commitChanges();
                    rec.requestFocus();
                    rec.resume();

                    System.out.println("Ready go...");
                    while(sp.equals(""));
                    return sp;
            }
            catch (Exception e)
            {
                    e.printStackTrace();
            }
            return sp;
        }
}

public class Splash extends Frame implements Runnable
{
        Image i1;
        Thread t1=null;
        public void start()
        {
            if(t1==null)
            {
                    t1=new Thread(this);
                    t1.start();
            }
        }
```

```
    public void stop()
    {
        if(t1!=null)
        {
            t1.stop();
            t1=null;
        }
    }
    public void run()
    {
        try
        {
            Thread.sleep(4000);
            hide();
            Mainfrm dsp=new Mainfrm();
            dsp.show();
        }
        catch(InterruptedException ie){}
    }
    public Splash()
    {
        i1=getToolkit().getImage("Splash.jpg");
        setSize(700,500);
        setLocation(50,50);
        show();
        start();
    }
    public void paint(Graphics g)
    {
        g.drawImage(i1,100,100,this);
    }
}

class Mainfrm extends Frame implements ActionListener,Runnable
{
```

```
Image i1;
private String str;
private MenuBar mb;
private Menu Masters,Operations,Quering,Reports,Help,qdisease;
private MenuItem discate,disinfo,patients,symptoms,memview;
private MenuItem disidn,index,id3,id1,id2,id4,id5,exit;
private MenuItem qsymptoms;
private Thread t1;
private SpeechToText m=new SpeechToText();

public void paint(Graphics g)
{
        g.drawImage(i1,100,100,this);
}

public void actionPerformed(ActionEvent ae)
{
        MenuItem b1=(MenuItem)ae.getSource();
        setTitle(b1.getLabel());
        if(ae.getSource()==disinfo)
        {
                Listbooks lk=new Listbooks();
        }
        if(ae.getSource()==memview)
        {
                setTitle("Ledger view");
                try
                {
                        Runtime rt=Runtime.getRuntime();
                        rt.exec("bookview");
                }
                catch(IOException io){ }
        }
        if(ae.getSource()==id5)
        {
```

```
                {
                        Runtime rt=Runtime.getRuntime();
                        rt.exec("c:\\ViaVoice\\bin\\enroll.exe -L En_IN");
                        }catch(IOException io){}
                }
        }
        else if(ae.getSource()==discate)
        {
                try
                {
                        Runtime rt=Runtime.getRuntime();
                        rt.exec("bookpro");
                }catch(IOException io){}
        }
        else if(ae.getSource()==disidn)
        {
                try
                {
                        Runtime rt=Runtime.getRuntime();
                        rt.exec("bookview");
                }catch(IOException io){}
        }
public void start()
{
        if(t1==null)
        {
                m.Begin("Demo.gram");
                setTitle("Speech Recognition Started");
                t1=new Thread(this);
                t1.start();
        }
}

public void stop()
{
        if(t1!=null)
```

```
        {
                t1.stop();
                t1=null;
        }
}

public void run()
{
        while(true)
        {
                str=m.MainMethod();
                str=str.trim();
                setTitle(str);
                if(str.equalsIgnoreCase("add books"))
                {
                        setTitle("Please enter the details");
                        try
                        {
                                Runtime rt=Runtime.getRuntime();
                                rt.exec("bookpro");
                        }catch(IOException io){}
                }
                if(str.equalsIgnoreCase("book issue"))
                {
                        setTitle("Book issued details");
                        try
                        {
                                Runtime rt=Runtime.getRuntime();
                                rt.exec("bookview");
                        }catch(IOException io){}
                }
                if(str.equalsIgnoreCase("exit"))
                {
                        hide();
                        System.exit(0);

                }
```

```
                    try
                    {
                            Thread.sleep(100);
                    }
                    catch(InterruptedException ie){}
            }
    }
```

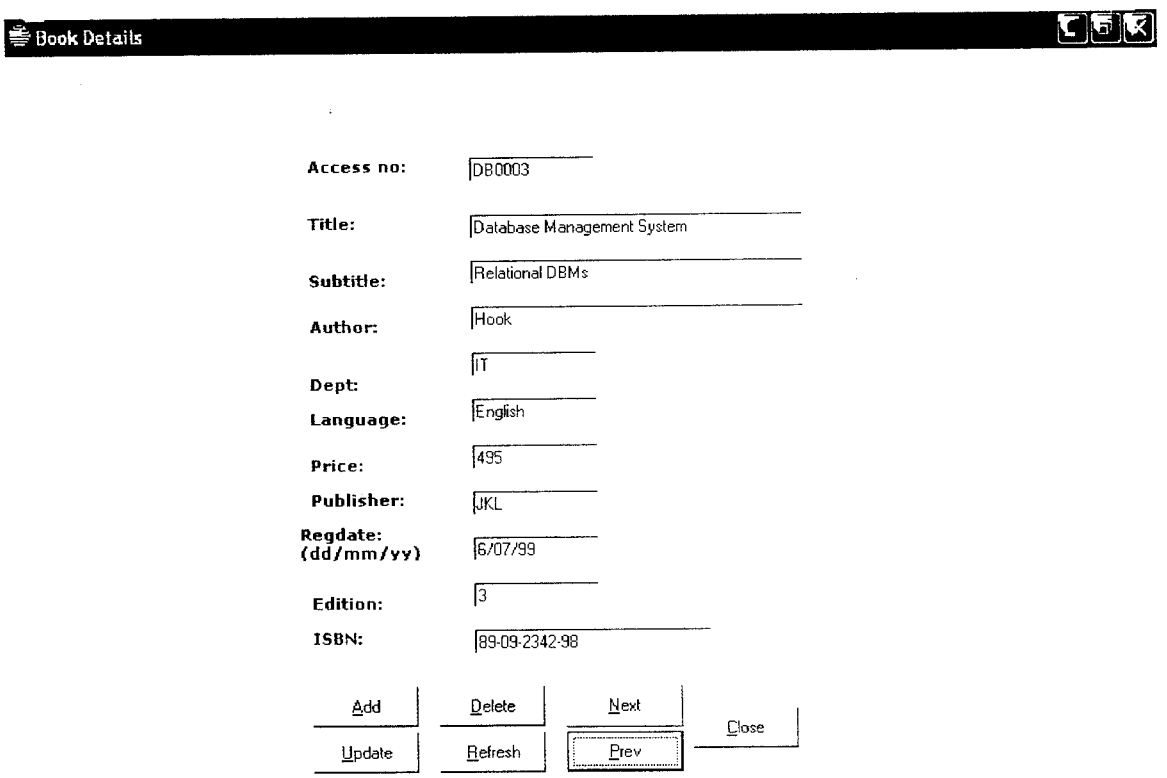
```
//USER VALIDATION
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.util.*;
import java.lang.*;
import java.sql.*;

public class LogonDialog extends Frame implements ActionListener
{
        private JTextField jtext;
        private JPasswordField jpass;
        private JLabel jl1,jl2;
        private JButton jb1,jb2;

        public void dbConnection(String uid,String pwd)
        {
                try
                {
                        Driver drv=new sun.jdbc.odbc.JdbcOdbcDriver();
                        Connection
                       conn=DriverManager.getConnection("jdbc:odbc:netdb");
                        Statement stmt=conn.createStatement();
                        ResultSet rs=stmt.executeQuery("select * from Users");
                        String u,p;
                        boolean flag1=false;
                        boolean flag2=false;
```

```
                ResultSet rs=stmt.executeQuery("select * from Users");
                String u,p;
                boolean flag1=false;
                boolean flag2=false;
                while(rs.next())
                {
                        u=rs.getString(1);
                        if(u.equalsIgnoreCase(uid))
                        {
                                flag1=true;
                                if(p.equals(pwd))
                                {
                                        flag2=true;
                                        break;
                                }}
                }
                if(flag1==true && flag2==true)
                {
                        setTitle("Entering");
                        hide();
                         Mainfrm obj=new Mainfrm();
                        obj.setSize(800,600);
                        obj.setVisible(true);
            }
        }
}
```

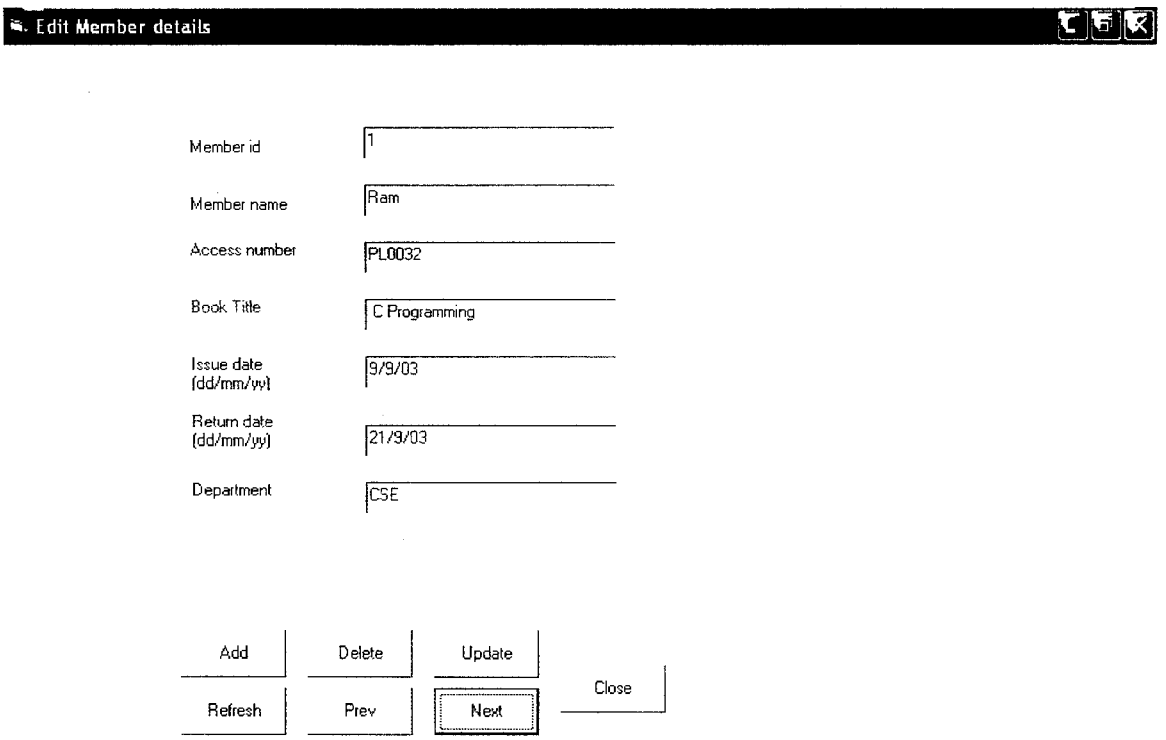
SAMPLE SCREEN

INPUT SCREEN

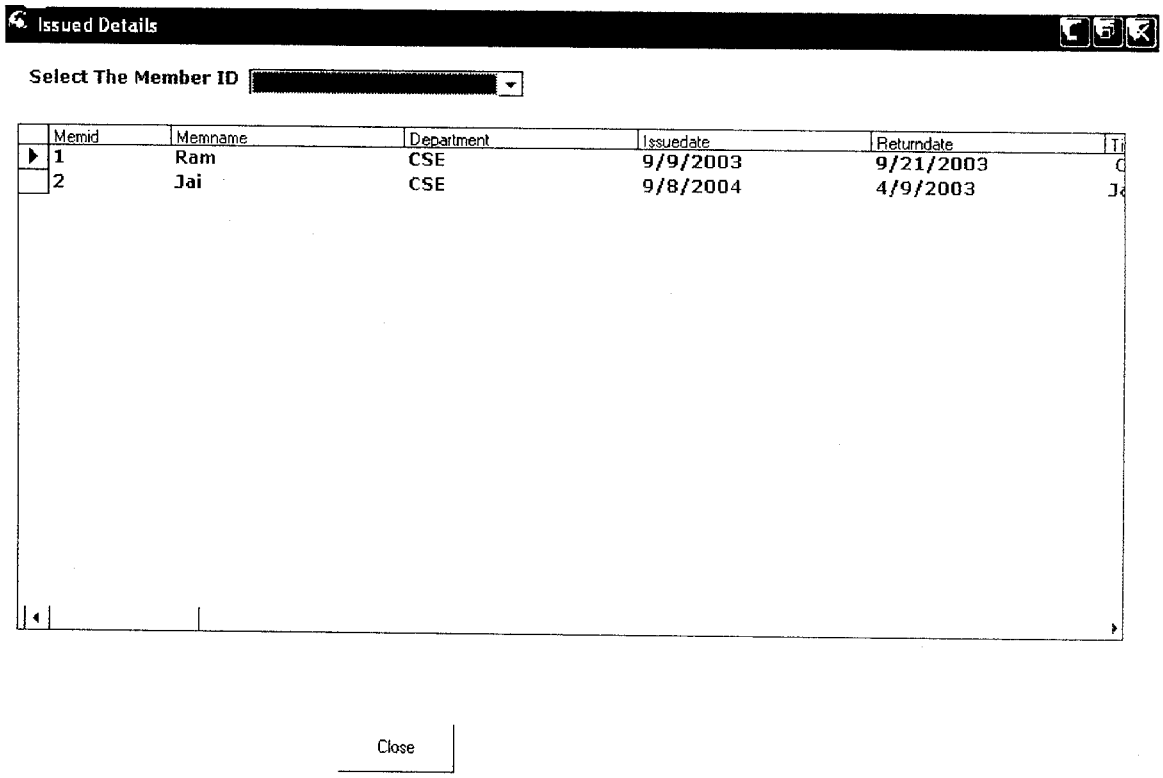
ADD AND VIEW BOOK DETAILS



EDIT & VIEW MEMBER DETAILS

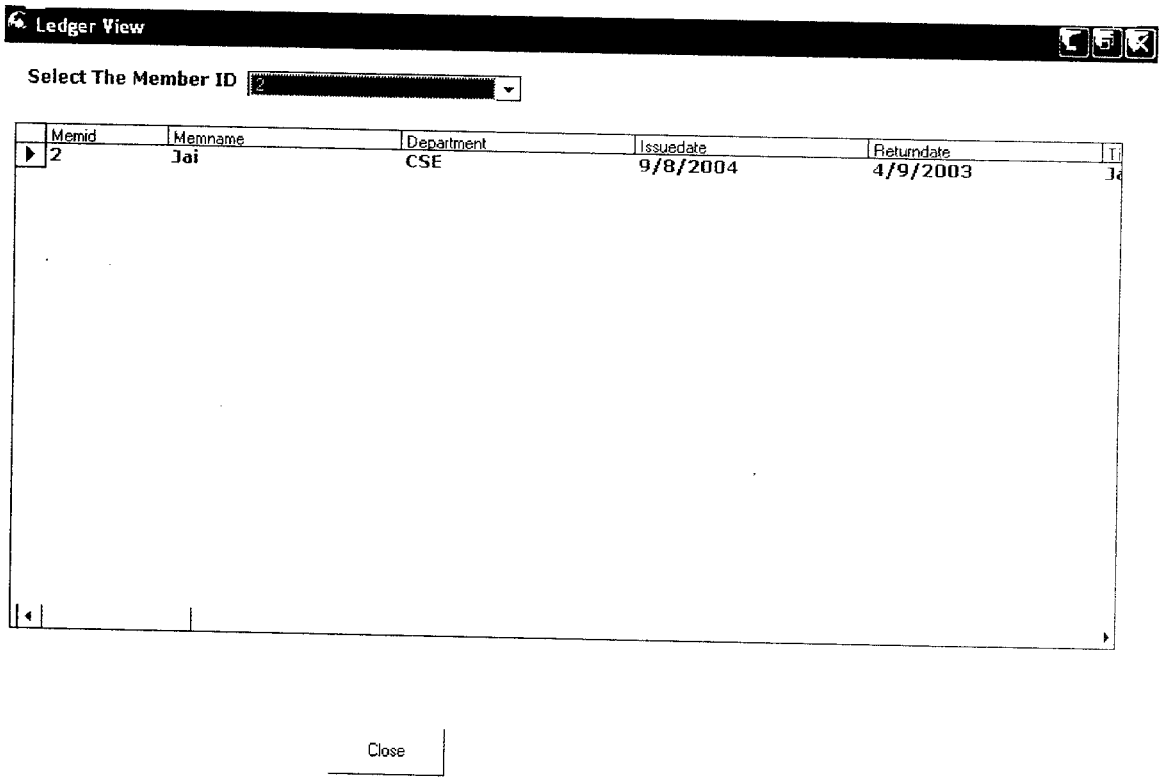


LEDGER VIEW

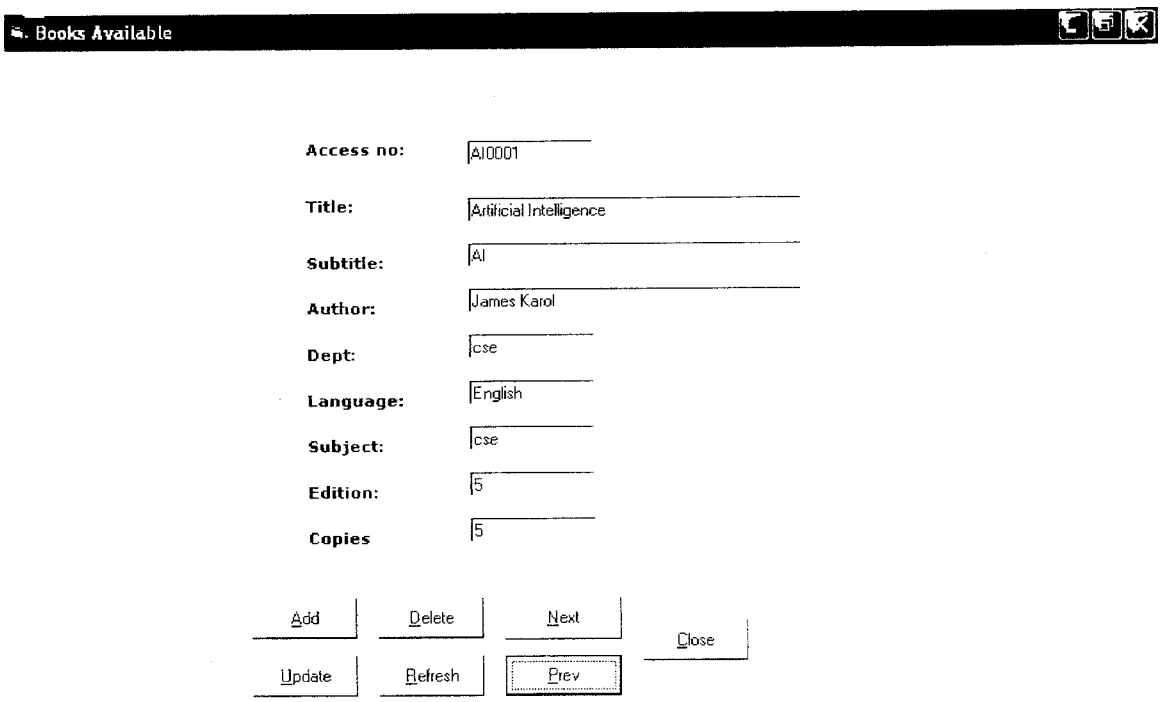

Issued Details
Select The Member ID
Memid
Memname
Department
Issuedate
Returndate
1
Ram
CSE
9/9/2003
9/21/2003
2
Jai
CSE
9/8/2004
4/9/2003
Close

OUTPUT SCREEN

INDIVIDUAL MEMBER ACCOUNT



BOOKS AVAILABILITY



EXCEPTION HANDLING

UNAVAILABILITY EXCEPTION