



# **KERNEL BASED DESKTOP CONTROLLER**

**Via VOICE**

*P-1596*

**A PROJECT REPORT**

*Submitted By*

**P.ANNAPOORANI**

**71202104002**

**N.MEENAKSHI**

**71202104020**

*in partial fulfillment for the award of the degree*

*of*

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE**

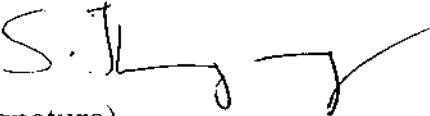
**ANNA UNIVERSITY: CHENNAI 600025**

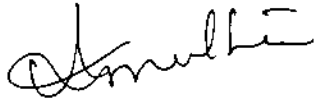
**APRIL 2006**

**ANNA UNIVERSITY:CHENNAI 600025**

**BONAFIDE CERTIFICATE**

Certified that this project report “**Kernel Based Desktop Controller via Voice**” bona fide work of “P.Annapoorani (71202104002) and N.Meenakshi (71202104020)”, who carried out the project work under my supervision.

  
(Signature)

  
(Signature)

Dr.S.Thangasamy

Ms. Amutha Venkatesh

Head of the Department

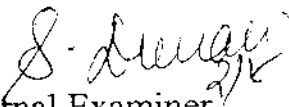
Supervisor

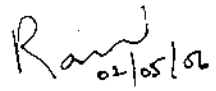
Senior Lecturer

Dept. of Computer Science & Engg.,  
Kumaraguru College of Technology,  
Chinnavedampatti P.O.,  
Coimbatore-641006.

Dept. of Computer Science & Engg.,  
Kumaraguru College of Technology,  
Chinnavedampatti P.O.,  
Coimbatore-641006.

Submitted for Viva Voce Examination held on 2/5/06

  
Internal Examiner

  
External Examiner

# DECLARATION

We hereby declare that the project entitled “**Kernel Based Desktop Controller via Voice**” is a record of the original work done by us and to the best of our knowledge.

The report is submitted in partial fulfillment of the requirements for the award for the Degree of Bachelor of Computer Science and Engineering of Anna University, Chennai.

Place: Coimabtoe.

Date : 26.4.06

*P. Annapoorani*

(P. Annapoorani)

*N. Meenakshi*

(N. Meenakshi)

# ACKNOWLEDGEMENT

We would like to extend our gratitude to the Principal of our college, Dr. K. K. Padmanaban for providing us the required resources to proceed with our project.

We would like to express our sincere thanks to our Head of the Department, Dr.S.Thangasamy for his guidelines that motivated us to develop a good product.

We are very grateful to our Project coordinator, Ms.P.Devaki, Assistant Professor, for her constant advice and support during the project.

We take immense pleasure in expressing our heartfelt thanks to our guide Ms. Amutha Venkatesh for her inspiration and guidance throughout this project. We are very grateful to her for her help rendered to us in handling various tough spots during this project.

We would like to express our thanks to all the members of the faculty of the Department of Computer Science & Engineering for their support. We also extend our gratitude to all the lab technicians who had helped us during the course of our project.

Last but not the least we would like to extend our heartfelt thanks to our parents and our brothers and sisters and friends for their encouragement and support. Also we thank all those who have helped us directly or indirectly in our project.

## *ABSTRACT*

# ABSTRACT

“The Kernel Based Desktop Controller via Voice” project is aimed at providing software to control the actions of the desktop using the user’s voice alone. The various applications on the desktop are to be invoked by this software by correctly identifying the words spoken by the user and converting them into commands which can invoke the desired event or application. The entire desktop of the computer, starting from ‘my computer’ to ‘start button’ is controlled. The spoken words are inputted to the system in the form of voice and converted in to commands by the system proposed.

This system consists of the interface which gets the input from the user. The voice of the user is got through the microphone. It is then sent to the Speech recognition engine present in the Microsoft SDK 5.1. The voice is converted to text by the SDK. The text is then sent to the program, which refers the XML files for the grammar. In the XML file, we define the tokens that are to be sent to the program. Depending on the tokens, corresponding events are invoked. To verify whether the computer has correctly converted the speech, the program sends the converted text to the TTS engine and it also displays the text.

# TABLE OF CONTENTS

CHAP.NO.	TITLE	PAGE NO.
	ACKNOWLEDGEMENT	iv
	ABSTRACT	vi
	LIST OF FIGURES	x
	LIST OF SYMBOLS, ABBREVIATIONS AND NOMENCLATURE	xi
	LIST OF TABLES	xii
1	INTRODUCTION	2
	1.1 SPEECH FORMS TECHNOLOGY	3
	1.2 MICROSOFT SPEECH SDK 5.1	4
	1.2.1 INTRODUCTION TO SAPI 5	4
	1.3 EXISTING SYSTEM	5
	1.4 PROPOSED SYSTEM	5
	1.4.1 PROPOSED SYSTEM'S APPLICATIONS	5
	1.4.2 PROPOSED SYSTEM'S ADVANTAGES	5
	1.4.3 PROPOSED SYSTEM'S DRAWBACKS	6
2	OPERATING ENVIRONMENT	8
	2.1 HARDWARE REQUIREMENTS	8
	2.2 SOFTWARE REQUIREMENTS	8
	2.3 OPERATING SYSTEM	8
	2.4 DESIGN AND IMPLEMENTATIONS	9
	CONSTRAINTS	
	2.5 USER DOCUMENTATION	9
	2.6 ASSUMPTIONS AND DEPENDENCIES	9

CHAP.NO	TITLE	PAGE NO.
3	SOFTWARE REQUIREMENTS SPECIFICATION	11
	3.1 INTRODUCTION	11
	3.2 OVERALL DESCRIPTION	11
	3.2.1 PRODUCT PERSPECTIVE	11
	3.2.1.1 PRODUCT FEATURES	11
	3.2.1.2 USER CLASSES AND CHARACTERISTIC	12
	3.3 SYSTEM FEATURES	12
	3.3.1 DESIGNING OF FORMS	12
	3.3.2 VOICE TRAINING AND RECOGNITION	13
	3.3.3 GRAMMAR FILE	13
	3.4 EXTERNAL INTERFACE REQUIREMENTS	14
	3.4.1 USER INTERFACE	14
	3.4.2 HARDWARE INTERFACE	14
	3.4.3 SOFTWARE INTERFACE	14
	3.4.4 COMMUNIOICATION INTERFACE	15
	3.5 OTHER NONFUNCTIONAL REQUIREMENTS	15
	3.5.1 PERFORMANCE REQUIREMENTS	15
	3.5.2 SAFETY REQUIREMENTS	15
	3.6 SOFTWARE QUALITY ATTRIBUTES	15
	3.6.1 ADOPTABILITY	15
	3.6.2 AVAILABILITY	15
	3.6.3 INETROPERABILITY	16
	3.6.4 RELIABILITY	16



CHAP.NO	TITLE	PAGE NO
	3.6.5 REUSABILITY	16
	3.6.6 TESTABILITY	16
4	SYSTEM DESIGN	18
	4.1 PROPOSED SYSTEM ARCHITECTURE	18
	4.2 COLLOBORATION DIAGRAM	19
	4.3 SEQUENCE DIAGRAM	20
5	DETAILED DESIGN	22
	5.1 MODULE DESCRIPTION	22
	5.1.1 INTERACTION WITH THE USER	22
	5.1.2 SPEECH TO TEXT CONVERSION	26
	5.1.3 VOICE TRAINING AND RECOGNITION	26
	5.1.4 XML FILES	27
	5.2 DESKTOP CONTROLLING	30
	5.2.1 HOW TO START	30
	5.2.2 EXPANSION OF THE MENU	30
	5.2.3 HOW IT WORKS	33
	5.2.4 COMMANDS AVAILABLE	34
	5.2.5 SPEECH LIBRARY	35
	5.2.6 HOOKING MENU	37
6	FUTURE ENHANCEMENTS	39
7	CONCLUSION	41
8	APPENDIX	43
9	REFERENCES	51

# LIST OF FIGURES

**Page no.**

OVERVIEW OF SPEECH BASED HUMAN MACHINE INTERFACE	3
ARCHITECTURE OF THE SYSTEM	18
COLLABORATION DIAGRAM	19
SEQUENCE DIAGRAM	20
MAIN FORM	25
SYSTEM ARCHITECTURE OVERVIEW	26
USER INTERFACE	33
INPUT WINDOW	43
OUTPUT WINDOW	44

# **LIST OF SYMBOLS, ABBREVIATIONS AND NOMENCLATURE**

KBDCvV	- Kernel Based Desktop Controller via Voice
ASR	- Automated Speech Recognition
DLL	- Dynamic Link Library
TTS	- Text To Speech
SDK	- Software Development Kit
SAPI	- Speech Application Protocol Interface
COM	- Component Object Model
XML	- Extended Markup Language

# LIST OF TABLES

LIST OF COMMANDS AVAILABLE	34
----------------------------	----

## *INTRODUCTION*

# 1. INTRODUCTION

The Speech Recognition systems allow people to control a computer by speaking to it through a microphone, either entering text, or issuing commands to the computer, e.g. to load a particular program, or to print a document.

**KERNEL BASED DESKTOP CONTROLLER via VOICE (KBDCvV)** – A Computer Program for the Study of Natural Language and Communication between Man and Machine

- This is a stepping stone in ARTIFICIAL INTELLIGENCE
- It makes certain kinds of natural language conversation between man and computer possible.
- Input is analyzed and they are converted to text. Only the given particular set of keywords is processed.
- Responses are generated by invoking the corresponding DLL.

The system is designed to control the Desktop by activating programs by voice. The entire desktop of the computer, starting from my computer to start button is controlled. The spoken words are inputted to the system in the form of voice and converted in to commands by the system proposed.

## 1.1 SPEECH FORMS TECHNOLOGY

Speech is a sequence of structured sounds forming words and sentence. The speaker converts information that he wants to communicate to the machine into speech.

Speech then propagates as a sound wave to a microphone which is the entrance automatic speech recognition systems.

A microphone generates a signal that is decoded by ASR systems. The linguistic information that is encoded by speaker during speech production is extracted as a sequence of words.

The machine can further interpret the word sequence and perform the appropriate action.

Speech messages can be produced with a TTS and emitted towards with the speaker. Here the acoustic wave given is measured and analyzed in order to extract the linguistic information and derive a sequence of words.

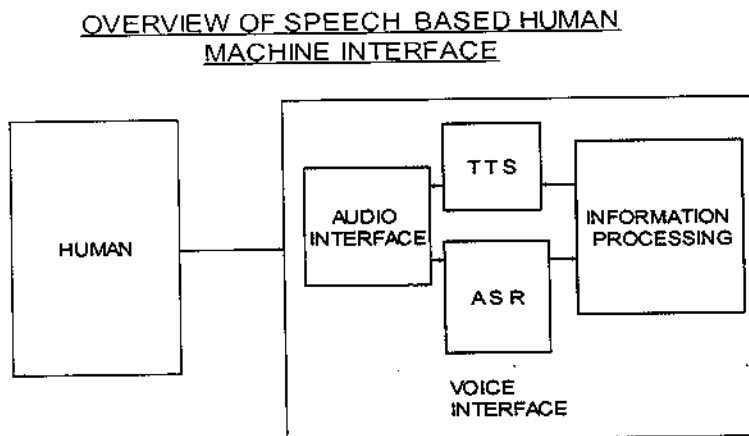


Fig 1.1

## **1.2 MICROSOFT SPEECH SDK 5.1**

Microsoft speech SDK 5.1 is a software development kit for building speech engines and applications for Microsoft windows. It has been designed primarily for the desktop speech developer. The SDK contains the Microsoft speech application programming interface the SAPI, the Microsoft continuous speech recognition engine and Microsoft concatenated speech synthesis (or text to speech) engine, a collection of speech oriented development tools for compiling source code and executing command and documentation on the most important SDK features.

### **1.2.1 INTRODUCTION TO SAPI 5**

SAPI model is based on the Component Object Model (COM) interface and Microsoft has defined two distinct levels of SAPI services:

- High level SAPI provides “command and control” level of services. This is good for detecting menu and system level commands and for speaking simple text.
- Low level SAPI provides the much more flexible interface and allows programmer’s access to extended speech recognizer and text to speech services.



## **1.3 EXISTING SYSTEM**

The Existing system runs the executable file directly without activating the kernel controls. Hence the area of application is restricted. It is platform dependent. Operations done in this system are complex. Speech soft wares available as of now are expensive

## **1.4 PROPOSED SYSTEM**

In the proposed system, the voice of the user is got through the microphone. It is then sent to the Speech recognition engine present in the Microsoft SDK 5.1. The voice is converted to text by the SDK. The text is then sent to the program, which refers the XML files for the grammar. In the XML file, we define the tokens that are to be sent to the program. Depending on the tokens, corresponding events are invoked. To verify whether the computer has correctly converted the speech, the program sends the converted text to the TTS engine and it also displays the text.

The proposed system developed is financially feasible when compared with the existing system. The startup cost is high for the existing system when compared to the proposed system. It is simple and user friendly. Controls in the operating systems are directly used.

### **1.4.1 Application of proposed system**

- This is specially used for the computer users who are physically handicapped.
- Without the mouse, the entire desktop operations can be controlled.

### **1.4.2 Advantages of proposed system**

- Accessible from remote areas with the help of microphones.
- Platform-independent as .NET-framework is used.

### **1.4.3 Drawbacks**

- It is very difficult to restrict the speech library.
- Difficult to change or control the OS instead of kernel.
- Speech SDK can give only appropriate result with no accuracy.

# OPERATING ENVIRONMENT

## **2. OPERATING ENVIRONMENT**

### **2.1 HARDWARE REQUIREMENTS**

- Pentium IV processor
- 256 MB RAM
- Multimedia microphone
- Speakers

### **2.2 SOFTWARE REQUIREMENTS**

- The System needs “SPEECH APPLICATION PROGRAMMING INTERFACE 5” (SAPI 5).
- Microsoft Engine for English.
- Microsoft Speech SDK 5.1

### **2.3 OPERATING SYSTEM**

- Windows XP processor
- Microsoft window 2000
- Microsoft window millennium
- Microsoft NT4.0 workstation or server and higher versions.

## **2.4 DESIGN AND IMPLEMENTATION CONSTRAINTS**

The following is the listing of the various design constraints related to the product under development to function.

- The system can only run on the stand-alone computers, where the system to run on a network is considered to be an enhancement.
- The system must be installed with Microsoft Speech SDK 5.1 and SAPI5.
- The system must be equipped with a Microphone so that the user may be able to command the system to execute the various keyboard and mouse functions.

## **2.5 USER DOCUMENTATION**

The Desktop Controller is designed for a single user who is the Operator. The necessary documentation is provided to the Operator for using the system.

## **2.6 ASSUMPTIONS AND DEPENDENCIES**

It is assumed that the voice of the user is familiar to the system so that the system recognizes the user's voice commands. The system depends on the Microsoft Speech SDK5.1 and Speech Application Programming Interface (SAPI5).

# *SOFTWARE REQUIREMENTS SPECIFICATION*

## **3. SOFTWARE REQUIREMENTS SPECIFICATION**

### **3.1 INTRODUCTION**

This Software Requirement Specification describes the function and performance requirements of Kernel Based Desktop Controller via Voice. This is a system that is used to execute all the Microsoft Windows command and work with all applications of Microsoft Windows using voice. The objective of the system is to minimize the work of the user by doing all the work of keyboard and mouse through voice.

### **3.2 OVERALL DESCRIPTION**

#### **3.2.1 PRODUCT PERSPECTIVE**

The proposed work includes adding “Speech Recognition Capability” to the system. The system should be able to recognize the commands issued as voice. The Speech Recognition Engine that is attached to the system will convert the speech into text, thus providing suitable commands for execution purposes.

##### **3.2.1.1 PRODUCT FEATURES**

The system is capable of recognizing human voices as commands and matches it with an existing command. Once the command is recognized the system produces the desired result.

### **3.2.1.2 USER CLASSES AND CHARACTERISTICS**

Desktop controlling through voice is being designed for single user. This general user shall be referred to as user. The user has the access rights to all the functionality of the system.

## **3.3 SYSTEM FEATURES**

The modules involved in the project are :-

### **3.3.1 DESIGNING OF FORMS**

#### **3.3.1.1 Description and priority**

The forms in the system are designed using .NET framework. This is basically the front end of the product. This is medium priority module as the front end can change with the wide variety of applications.

#### **3.3.1.2 Stimulus/response Sequences**

The user can either activate or deactivate the software. If the software is activated then the user can control all the desktop activities using his voice. The mic volume and the accuracy level can be viewed.

#### **3.3.1.3 Functional requirements**

The .NET Framework is used for designing the forms. The system first converts the command in the form of voice into text which requires Microsoft Speech SDK5.1 and SAPI. The user should be informed about the command that is executed by Microsoft Agent for better interaction of the user with the system.



## **3.3.2 VOICE TRAINING AND RECOGNITION**

### **3.3.2.1 Description and Priority**

The user when using the system for the first time has to train the system so that it could recognize the users' voice modulation. Once trained the system should recognize the commands given by the user.

### **3.3.2.2 Stimulus/Response Sequences**

The input to this module is given by the user through microphone. The speech is converted to text and the command is recognized for execution.

### **3.3.2.3 Functional requirements**

The user can start interacting with the system by saying the "Activate" command to it. The system would start listening to the user for the commands that are said to it for execution. Speech Application Programming Interface (SAPI) is required for matching the commands given by the user with the commands that the system recognizes.

## **3.3.3 GRAMMAR FILE**

### **3.3.3.1 Description and Priority**

The commands are stored in an XML file and the input from the user is matched with the available commands in the file and the execution is done. This module has a high priority, as it is the core functionality of the product.

### 3.3.3.2 Stimulus/Response Sequences

The input from the user in the form of speech is converted into text and then the text is matched with the commands already available in the system for execution.

### 3.3.3.3 Functional requirements

XML is required for storing the commands.

## **3.4. EXTERNAL INTERFACE REQUIREMENTS**

### **3.4.1 USER INTERFACE**

The user interface will consist of forms. It is very user friendly and the user can easily understand all the options available.

### **3.4.2 HARDWARE INTERFACE**

Microphone is used to obtain the command from the user in the form of speech. The user can dictate the commands directly to the system for execution, by using his/her voice. The system can understand the commands and produce the relevant function.

### **3.4.3 SOFTWARE INTERFACES**

The Speech Application Programming Interface converts the speech into text format. The C# .NET program compares the text with the commands stored in the XML file and the execution is done.

### **3.4.4 COMMUNICATION INTERFACES**

The user Communicates with the system using the microphone to train the system to recognize his voice modulation. Once the system is trained, the user can easily interact with the system and can do the entire keyboard and mouse functions by using only his/her voice.

## **3.5 OTHER NONFUNCTIONAL REQUIREMENT**

### **3.5.1 PERFORMANCE REQUIREMENTS**

The performance of the system is based upon the training that the user gives to the product.

### **3.5.2 SAFETY REQUIREMENTS**

This product is very safe to use and does not corrupt any files. Anyone can use the software and there is no restriction.

## **3.6 SOFTWARE QUALITY ATTRIBUTES**

### **3.6.1 ADAPTABILITY**

This product works only with in Microsoft Windows Applications. It works with any Microsoft Windows Operating Systems.

### **3.6.2 AVAILABILITY**

Availability of the product will be dependent upon the operation hours of any computer system that it is installed and the availability of the

speech engine. As such, it must be accessible at every hour on any type of computer system.

### **3.6.3 INTEROPERABILITY**

The product is platform independent. It works with any Microsoft platform

### **3.6.4 RELIABILITY**

The reliability of the software is based on the training given by the user. The overall reliability of the product is predicted to be around 60-70 percent.

### **3.6.5 REUSABILITY**

The C# coding could be used with different speech engines of different operating systems to change the platform that it works in.

### **3.6.6 TESTABILITY**

The accuracy of the product can be tested by changing the voice of the user who trained it and by giving wrong commands

# *SYSTEM DESIGN*

---

## 4. SYSTEM DESIGN

### 4.1 PROPOSED SYSTEM'S ARCHITECTURE

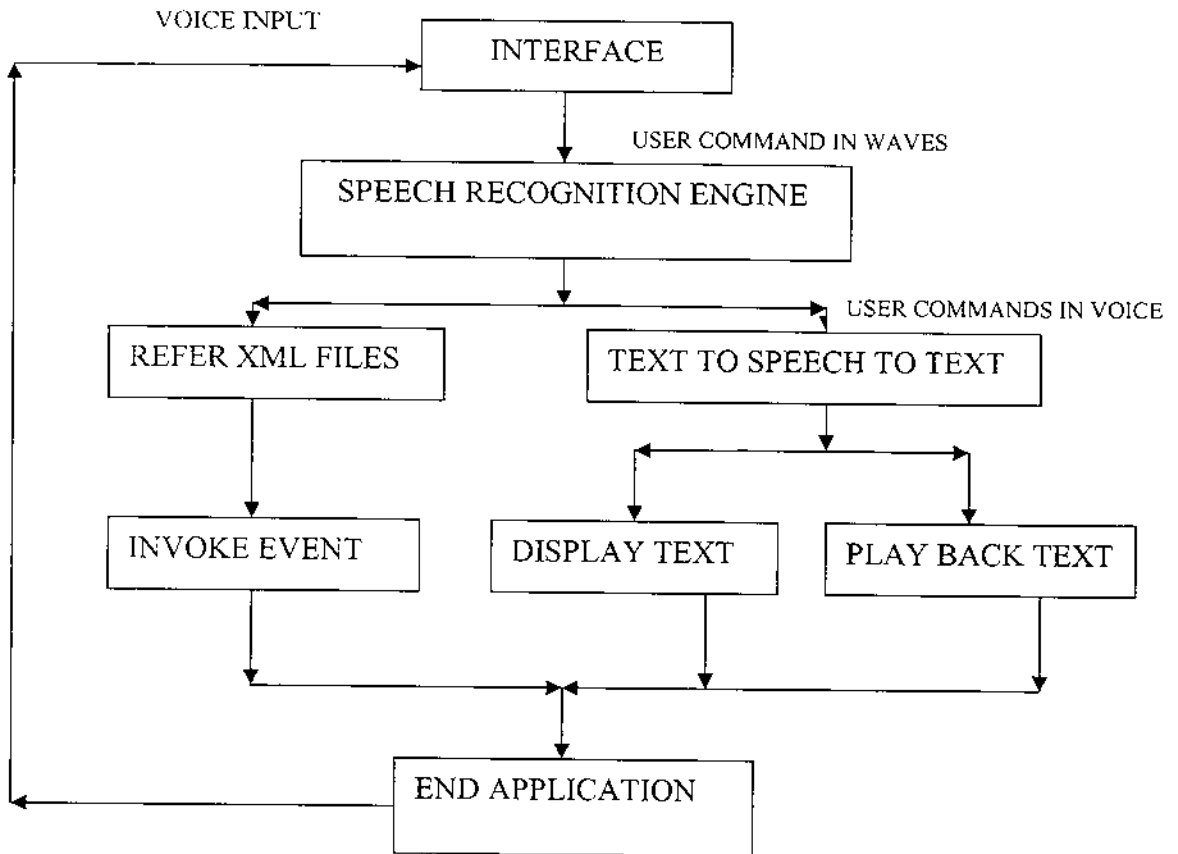
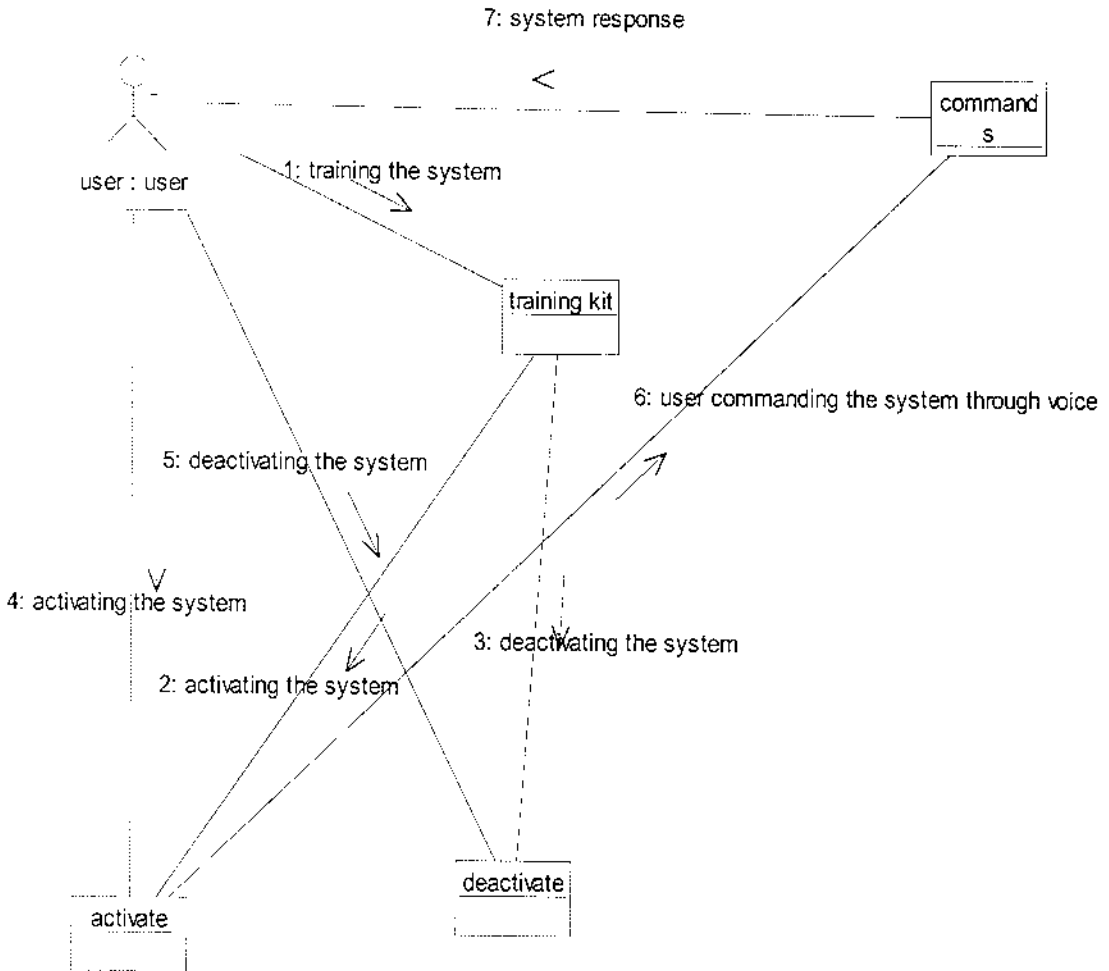


Fig 4.1

## 4.2 COLLABORATION DIAGRAM



**Fig 4.2** Collaboration Diagram

## 4.3 SEQUENCE DIAGRAM

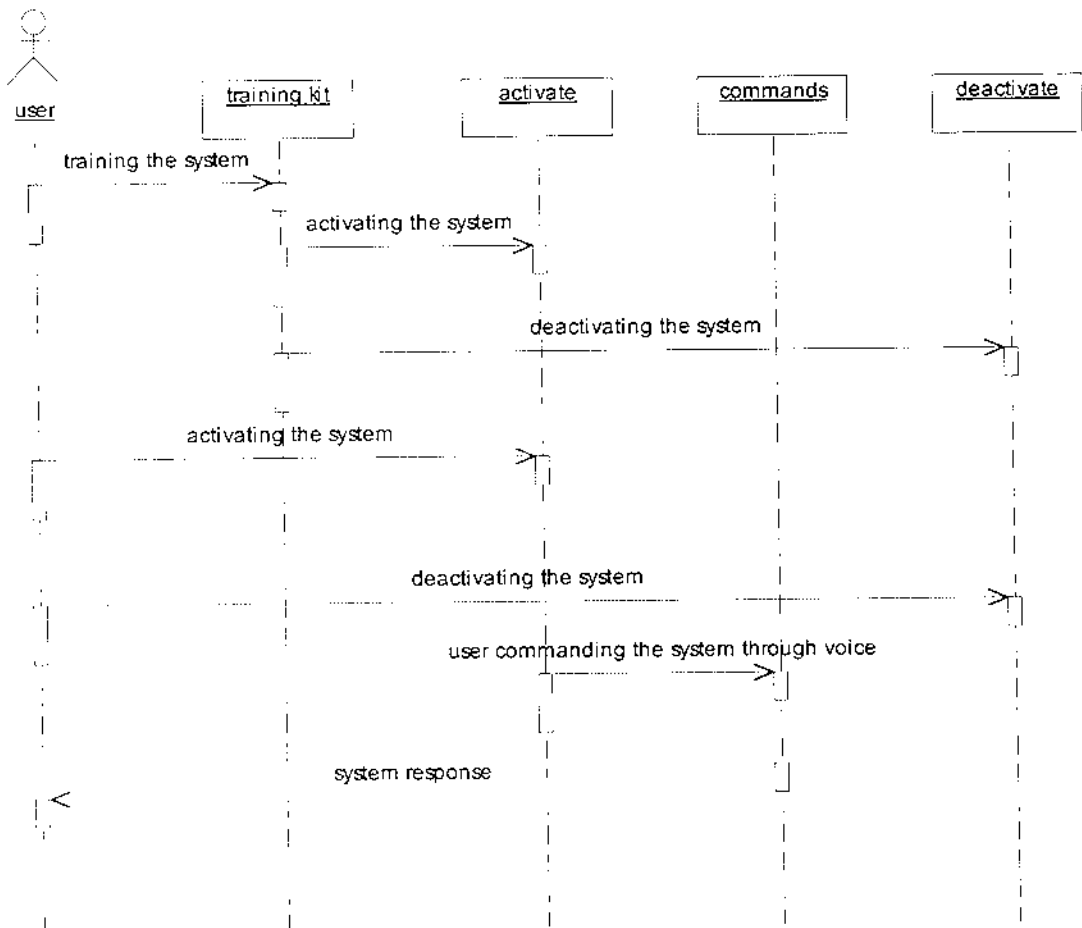


Fig 4.3 Sequence Diagram



## *DETAILED DESIGN*

## **5. DETAILED DESIGN**

### **5.1 MODULE DESCRIPTION**

The KBDCvV is logically divided into three modules as listed below.

#### **5.1.1 Interaction With The user**

The project allows the user to Activate or Deactivate the KBDCvV. Once activated the system could understand the Commands that are given by the user to perform the entire keyboard and mouse functions. To increase the efficiency of this system the user need to train the software and the system could be able to understand the users' commands more effectively and efficiently. The interface is very user friendly and the user could easily understand the operation of this system. The interface through which the user interacts with the program consists of 1.Welcome page 2.Command list 3.Accuracy 4.Profile Change 5.Favorites 6.Add favorites 7.About

#### **Main Form**

This is the form which takes care of the speech recognition and the conversion of voice to text. It integrates the various forms present in the application. This form also takes care of invoking the various applications on the desktop. Now let us see in detail about the various functions of this form. The DLL files required for capturing the system events are added in the main form. The objects required for the working of the speech recognition engine are created. The main form contains labels for the status

and accuracy level & a progress bar for mic volume. The menu items present in this form are start/stop listening, Mic training wizard, user training wizard, etc. In the form load event we initialize SAPI objects & load initial grammar. The commands of the menu items are added to the phrase list by capturing the menu items. The menu items are invoked and if they have a submenu then hook the sub menu. The main object RecoContext event is launched when engine recognizes a phrase. Depending on the phrase identified we take the necessary action. In this Recocontext event we calculate the accuracy and we dynamically change the maximum accuracy. We identify the various events using the rule name and not using the phrase. Whenever an event is to be occurred the corresponding file is included into the SAPI grammar and it is used for further references. The recognized words are displayed and they are played back using a Microsoft agent.

### **Favorites:**

The form contains list of the programs and their phrases. The list is generated each time by reading the XML file and hence it gets updated automatically whenever a new program is added through Add favorite command.

### **Add Favorites:**

The form contains two list boxes which displays the favorite programs and their corresponding phrases, two text boxes in which the program to be added and its phrase as favorite is written. The form also contains add, delete, save and browse buttons. When add button is clicked it is checked whether, the phrase given in the text box is already present in the list or not. If not present, then it is added to the list otherwise an error message is

displayed. The delete button deletes the particular item from the list box and its corresponding phrase by comparing the index of the selected item in the list box. The browse button allows the user to select the application by displaying a Open dialog box. The dialog box is invoked by using the Common Dialog Control. The selected application's name is placed in the text box along with its extension. The Save button enables the application to be actually added to the favorite list. First, the text boxes are checked for input and an error message is displayed whenever any one of the text boxes are empty. Then the given phrase is checked whether it is already present in the list box or not. If the phrase is not present then the application's name and the phrase is written into the XML file using the XML Writer. The application and the phrase is updated in the list box.

### **Commands List:**

The form contains the list of commands in a tree structure. It displays set of possible commands after a particular command is given. For e.g.: After the activate command, the possible commands like show about, show command list, deactivate, etc.

### **Accuracy Limit:**

The form contains the minimum accuracy level, above which the words spoken will be accepted as input. The incoming voice is compared with the vocabulary present and if the percentage of similarity is above the accuracy level it will be accepted.

## Profile Change:

It contains a list of the trained users of this application. The input of the user will be compared with the existing profiles and the profile which matches will be checked in the list.

If the current user wants to change the profile he can do so by manually selecting the required profile from the list and from then the input will be compared with that profile.

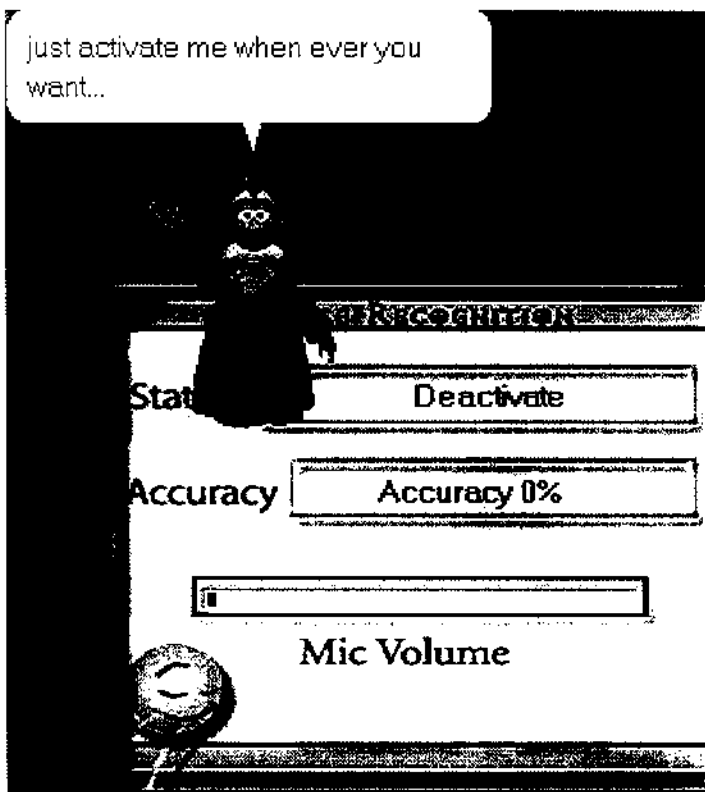
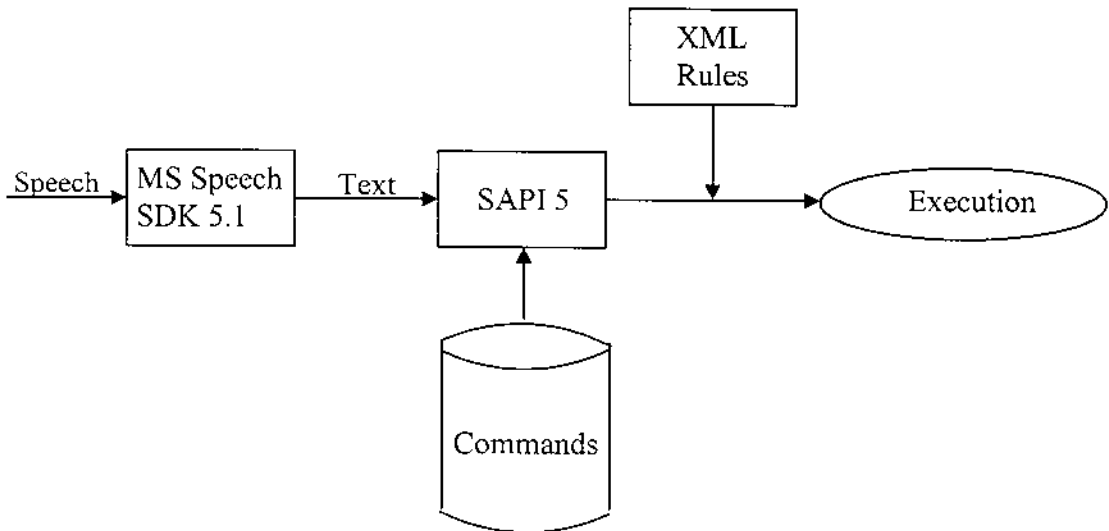


Fig 5.1  
Main Form

### 5.1.2 Speech To Text Conversion

The system is capable of interacting with the user by accepting all the Commands that is given to it by voice and executing the relevant function. To make this possible the input given by the user in the form of voice is converted into text by Microsoft Speech SDK5.1 and this text is matched with the default commands that the system can execute, if the commands match then the particular function for the command is performed by the system.



System Architecture overview

**Fig 5.2**

### 5.1.3 Voice Training and Recognition

The user needs to train the system before using this software, to improve the efficiency at which the system accepts the commands from the

user. The training is done so that the system can identify the users' voice modulation and run efficiently.

#### 5.1.4 Xml Files

The rules that are needed to be executed when a command is recognized are written in separate XML files. These files are matched with the respective commands by the Speech Application Programming Interface (SAPI 5). When the command is recognized by the system the respective XML file with the rules is executed by the program, to get the desired function to be performed.

An example of the XML code to activate the System is given below.

```
<! -- 409 = american english -->
<GRAMMAR LANGID="409">
<DEFINE>
<ID NAME="RID_Activate" VAL="0" />
<ID NAME="RID_CloseProgram" VAL="2" />
<ID NAME="RID_ShowAbout" VAL="3" />
<ID NAME="RID_ShowCommands" VAL="300" />
</DEFINE>
<RULE NAME="Activate" ID="RID_Activate"
  TOPLEVEL="ACTIVE">
  <O>please</O>
  <P>activate</P>
  <O>the</O>
  <O>computer</O>
</RULE>
```

```
<RULE NAME="CloseProgram" ID="RID_CloseProgram"
  TOPLEVEL="ACTIVE">
<P>close speech recognition</P>
</RULE>
<RULE NAME="ShowAbout" ID="RID_ShowAbout"
  TOPLEVEL="ACTIVE">
<P>about speech recognition</P>
</RULE>
<RULE NAME="ShowCommands" ID="RID_ShowCommands"
  TOPLEVEL="ACTIVE">
<P>commands list</P>
</RULE>
</GRAMMAR>
```

We define the following XML documents.

1. xmlactivate.xml
2. xmlcommands.xml
3. xmlalphabeticstate.xml
4. xmlnumericstate.xml
5. xmlfavorites.xml
6. xmlstart.xml
7. xmldeactivate.xml

### **xmlactivate.xml & xmldeactivate.xml**

These xml files define the grammar for activating or deactivating the application. It also makes the application to “start/stop listening” to the users voice.



### **xmlcommands.xml**

Here we define the actions to be taken for the different commands that are given by the user. Here we have to take care about the hierarchy in which the commands are given.

### **xmlalphabeticstate.xml**

In this file we define the grammar for the letters and commands that are present in the key board. This alphabetic state can be used when the user is using the text editor like notepad, MS Word etc.

### **xmlnumericstate.xml**

This file is similar to the xmlalphabeticstate.xml. But instead of the alphabets we have the grammar for the numbers. If we have the numbers in between the text we can to exit the alphabetic state and then switch to the numeric state.

### **xmlfavorites.xml**

Here we offer quick access to the frequently used applications specified in the favorites list in the interface

### **xmlstart.xml**

This xml file allows us to control the start menu present in the taskbar of our desktop.

## **5.2 DESKTOP CONTROLLING**

We planned to make some common tasks that every user does on his/her computer (opening/ closing programs, editing texts, calculating) possible not only by mouse/ keyboard but also by voice

### **5.2.1 How To Start**

In order to start talking right away you should do these two steps...

1. The first thing to do is adjusting the microphone by clicking the right mouse button and choosing the "Microphone training wizard"
2. The second thing to do is training the engine to your voice by choosing "user training wizard"

After these changes you have to make the program start listening again by clicking the right mouse button and choosing "Start listen". The more you train the engine, the better it will recognize your voice. We can see an improvement from the first training itself. After the program starts running it may be in a several "states", in every state it recognizes a list of specific commands.

### **5.2.2 A Expansion Of The Menu**

"Start listen"/"Stop listen" :

To enable/disable the mic (it's switched according to what you choose), after disabling the label's becomes red (accuracy and state) indicating our state.

"Use agent":

Though the agent is used only for giving feedback it's could be useful to know if your command is heard or not. You can disable it if you want or if you don't have an agent or if it is not working and still want to use the recognition. This also is being take care of, if the program didn't find the agent file or could not be loaded because of any other reason.

"Add favorites":

In the "activate" state you can say the command "*favorites programs*" and open a form with your favorites programs and running them by saying the program name. This menu will open a form showing your favorites programs so you can add/delete or edit them as you want.

"Change character":

This will allow you to change the agent character.

"Change accuracy limit":

The recognition accuracy of the software is displayed in the "Accuracy" label, you can choose this menu and change the accuracy limit that you want the program to respond to the command that it hears. We should do this to avoid responding to any voice or sound that he hears. You can rise this more every time that you train your computer and increase the recognition.

"Change user profile":

If the program is being used by several users you can choose to each user a profile and train the computer for each one (to add a user profile enter "control panel -> speech", here you can only choose existing one's).

"Mic training wizard...":

This is very important for improved recognition. The first thing to do in every computer is to activate this menu and setting up your mic or if you changed your mic to a new one.

"User training wizard...":

For a better recognition of the input given by the user we use this wizard.

### 5.2.3 How It Works

The initial state is in "deactivate" state that means the program is in sleepy state... After the command "activate" you will wake up the program and it start recognizes other commands.



User Interface

Fig5.3

For example "start" to activate the start menu, then you can say "programs" to enter the programs menu, from this point you can navigate by saying "down", " up", "right"... "OK" according the commands list. You can also say "commands list" from any point to see a form with the list of the commands that you can say.

One of the important states in the program is "menu" state, meaning that if a program is running (and focused) you can say "menu" to hook all menu items and start using them. For example if you are running notepad you could open new file by saying "menu"->"File"->"new file". Every time that you hook menu you could see how many menus the program hooked so you can start using them as commands.

Another nice state is "Numeric state", for example say the commands ... "favorites programs", "calculator", "enter numeric state", "one", "plus", "two", "equal" and see the result. Or you can open a site in "Alphabetic state", for example say the commands ... "favorites programs", "internet explorer", "enter alphabetic state", "menu", "down", "down", "OK", "enter alphabetic state", "c", "o", "d", "e", "dot", "c", "o", "m" and see the result.

## 5.2.4 Commands Available

**Table 4.1** Commands Available

<ul style="list-style-type: none"> <li>• deactivate           <ul style="list-style-type: none"> <li>○ close speech recognition</li> <li>○ about speech recognition               <ul style="list-style-type: none"> <li>▪ close   hide</li> </ul> </li> <li>○ activate               <ul style="list-style-type: none"> <li>▪ deactivate</li> <li>▪ up</li> <li>▪ down</li> <li>▪ right</li> <li>▪ left</li> <li>▪ enter   run   ok</li> <li>▪ escape   cancel</li> <li>▪ tab</li> <li>▪ menu   alt                   <ul style="list-style-type: none"> <li>○ All "activate" state + menu items</li> </ul> </li> <li>▪ start                   <ul style="list-style-type: none"> <li>○ deactivate</li> <li>○ up</li> <li>○ down</li> <li>○ right</li> <li>○ left</li> </ul> </li> </ul> </li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>• switch program           <ul style="list-style-type: none"> <li>○ tab   right</li> <li>○ shift tab   left</li> <li>○ enter   ok</li> <li>○ escape   cancel</li> </ul> </li> <li>• press key           <ul style="list-style-type: none"> <li>○ release   stop</li> <li>○ up</li> <li>○ down</li> <li>○ right</li> <li>○ left</li> </ul> </li> <li>• shut down           <ul style="list-style-type: none"> <li>○ right   tab</li> <li>○ left   shift tab</li> <li>○ escape   cancel</li> <li>○ enter   ok</li> </ul> </li> <li>• page up</li> <li>• page down</li> <li>• yes</li> <li>• no</li> </ul>
--	--

<ul style="list-style-type: none"> <li>○ enter   run   ok</li> <li>○ escape</li> <li>○ tab</li> <li>○ commands list</li> <li>○ programs</li> <li>○ documents</li> <li>○ settings</li> <li>○ search</li> <li>○ help</li> <li>○ run</li> <li>▪ commands list <ul style="list-style-type: none"> <li>○ close   hide</li> <li>○ page up</li> <li>○ page down</li> </ul> </li> <li>▪ close</li> <li>▪ favorites   favorites programs <ul style="list-style-type: none"> <li>○ close   hide</li> <li>○ A program name from the list</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>• enter numeric state <ul style="list-style-type: none"> <li>○ exit numeric</li> <li>○ state</li> <li>○ back   back space</li> <li>○ plus</li> <li>○ minus</li> <li>○ mul   multiply</li> <li>○ div   divide</li> <li>○ equal</li> <li>○ Numbers from 0 - 9</li> </ul> </li> <li>• enter alphabetic state <ul style="list-style-type: none"> <li>○ exit alphabetic state</li> <li>○ back space</li> <li>○ enter</li> <li>○ at ("@")</li> <li>○ underline ("_")</li> <li>○ dash ("-")</li> <li>○ dot (".")</li> <li>○ back slash ("/")</li> <li>○ Letters from A to Z</li> </ul> </li> </ul>
--	--

Table 5.1

## 5.2.5 Speech Library

Using SpeechLib;

When we activate the engine the initialization step takes place. There are mainly 3 objects involved:

1. An SpSharedRecoContext that starts the recognition process (must be shared so it can apply to all processes). It implements an ISpeechRecoContext interface. After this object is created we add the events we are interested in (in our case Audio Level and Recognition)
2. A static grammar object that can be loaded from XML file or programmatically implements ISpeechRecoGrammar the list of static recognizable words is shown in Fig 2 and attached for downloading dynamic grammar that lets adding rules implements ISpeechGrammarRule; the rule has two main parts :
  - o the phrase associated
  - o the name of the rule

Three basic functions that we will need are

- `initSAPI()` : To create grammar interface and activating interested events.
- `SAPIGrammarFromFile(string FileName)` : To load grammar from file.
- `SAPIGrammarFromArrayList(ArrayList PhraseList)` : To change grammar programmatically.

After initialization the engine still will not recognize anything until we load a grammar. There are two ways to do this: loading a grammar from file, or we can change the grammar programmatically.



## 5.2.6 Hooking Menus

When a program's menu is activated, by saying "Menu" its menu is hooked and its commands are added to the dynamic grammar. We used some unmanaged functions which we imported from *user32.dll*. The program also hooks the accelerators that are associated with each menu (that have & sign before them). The command is simulated with function `keybd_event` and is executed.

## *FUTURE ENHANCEMENTS*

## **6. FUTURE ENHANCEMENTS**

The future enhancements of this project would be used to perform all the advanced desktop operations like cut, copy and paste. It can be used to interpret directly with the text in the start menu. Application dependent software can be developed. Through speech recognition we can develop software to control the entire network by restricting the remaining channel. It can be further improved to provide security for a particular user.

We can further enhance this project to recognize phrases, interpret their meaning and function accordingly. We can also make the software to type the sentences directly into the word processor.

## *CONCLUSION*

## **7. CONCLUSION**

The Desktop Controlling through voice system was successfully developed in an effective and user-friendly approach. The software development process imparted in materializing this project strictly followed IEEE standards in all phases.

## APPENDIX

# 8. APPENDIX

## SCREEN SHOTS

### INPUT WINDOW

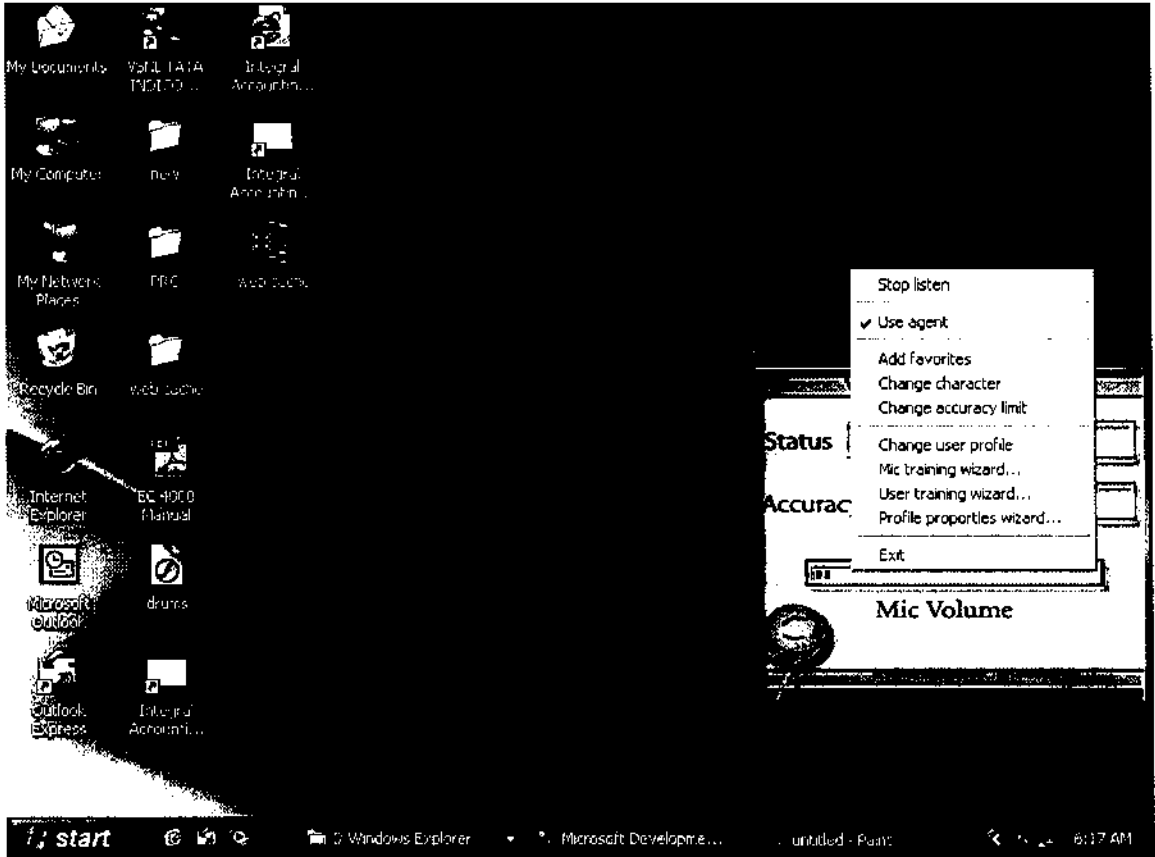


Fig A.1

# OUTPUT WINDOW

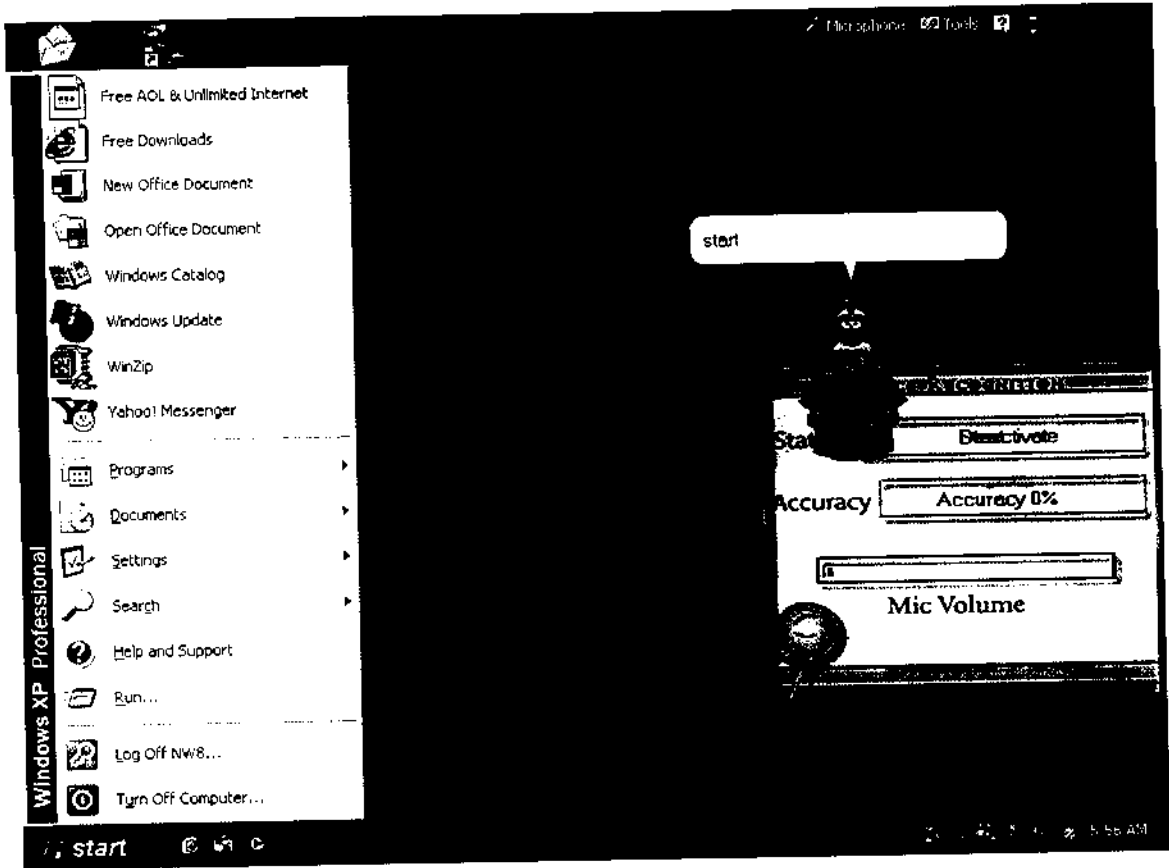


Fig A.2



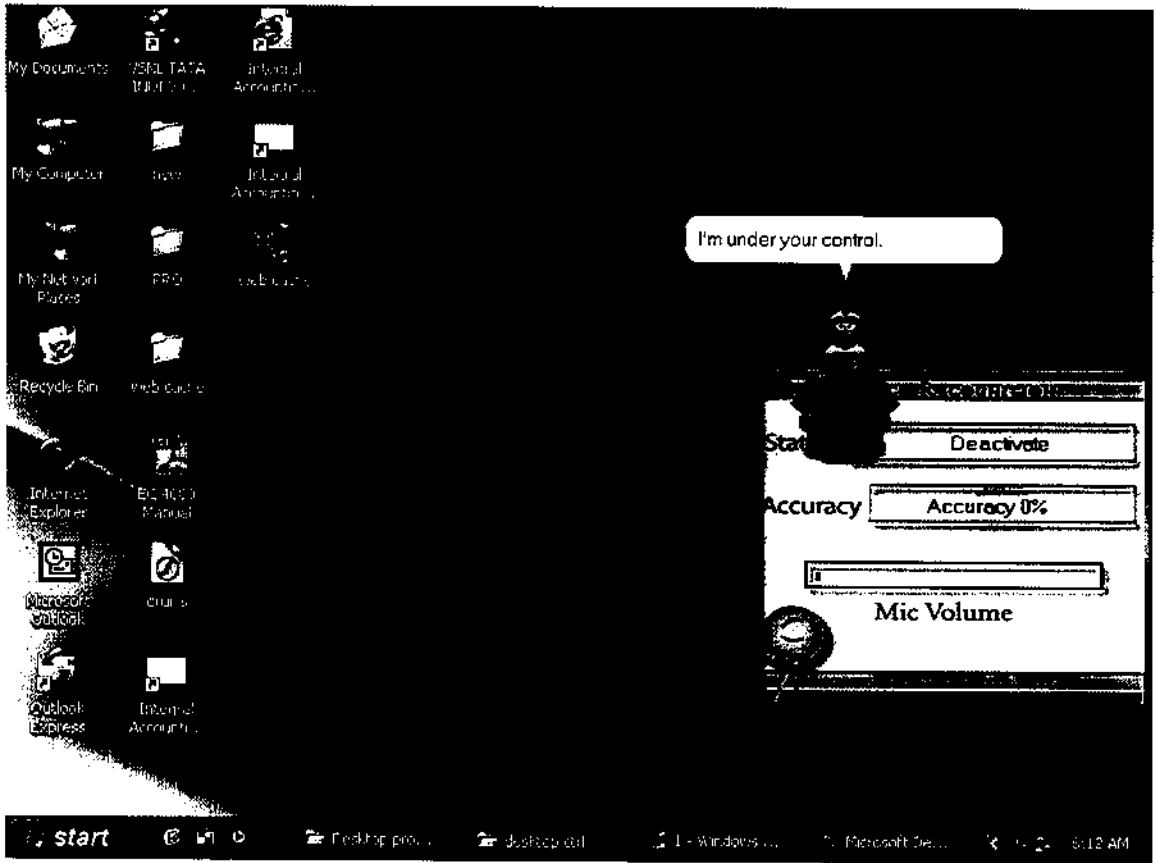


Fig A.3

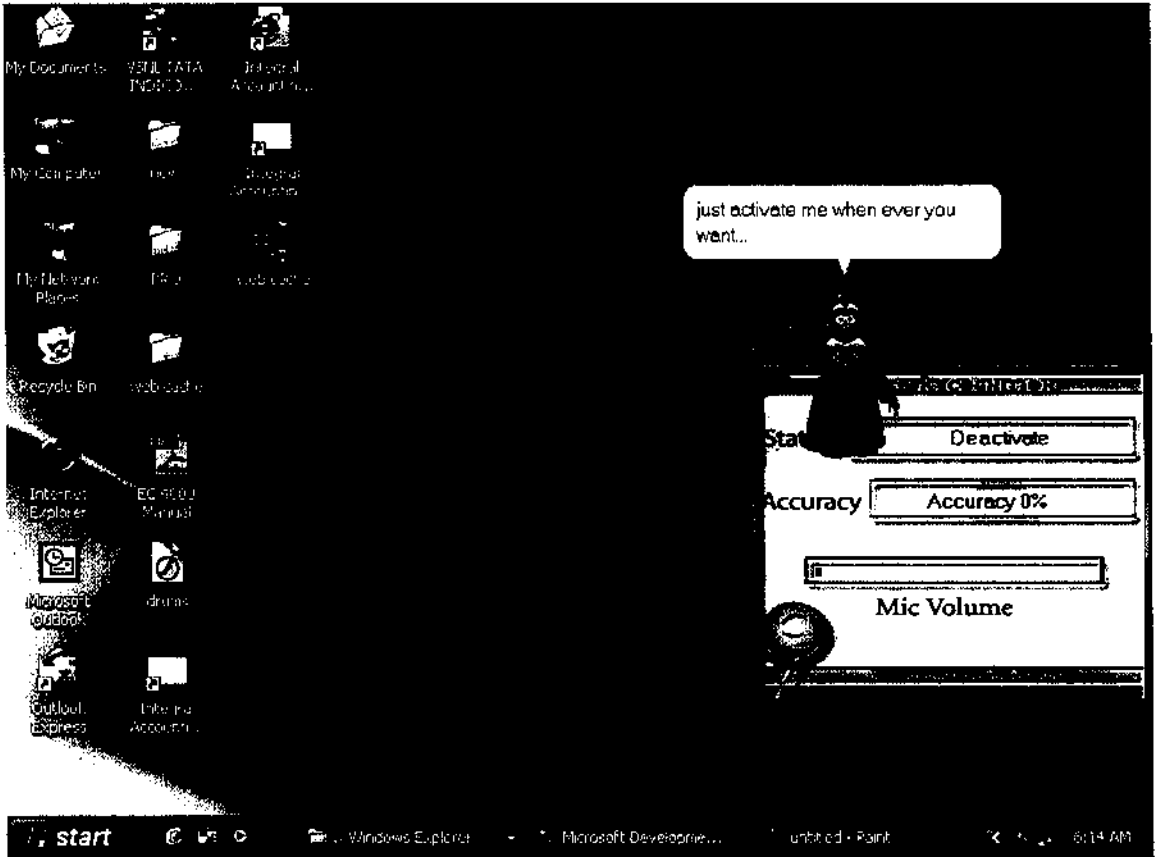


Fig A.4



Fig A.5

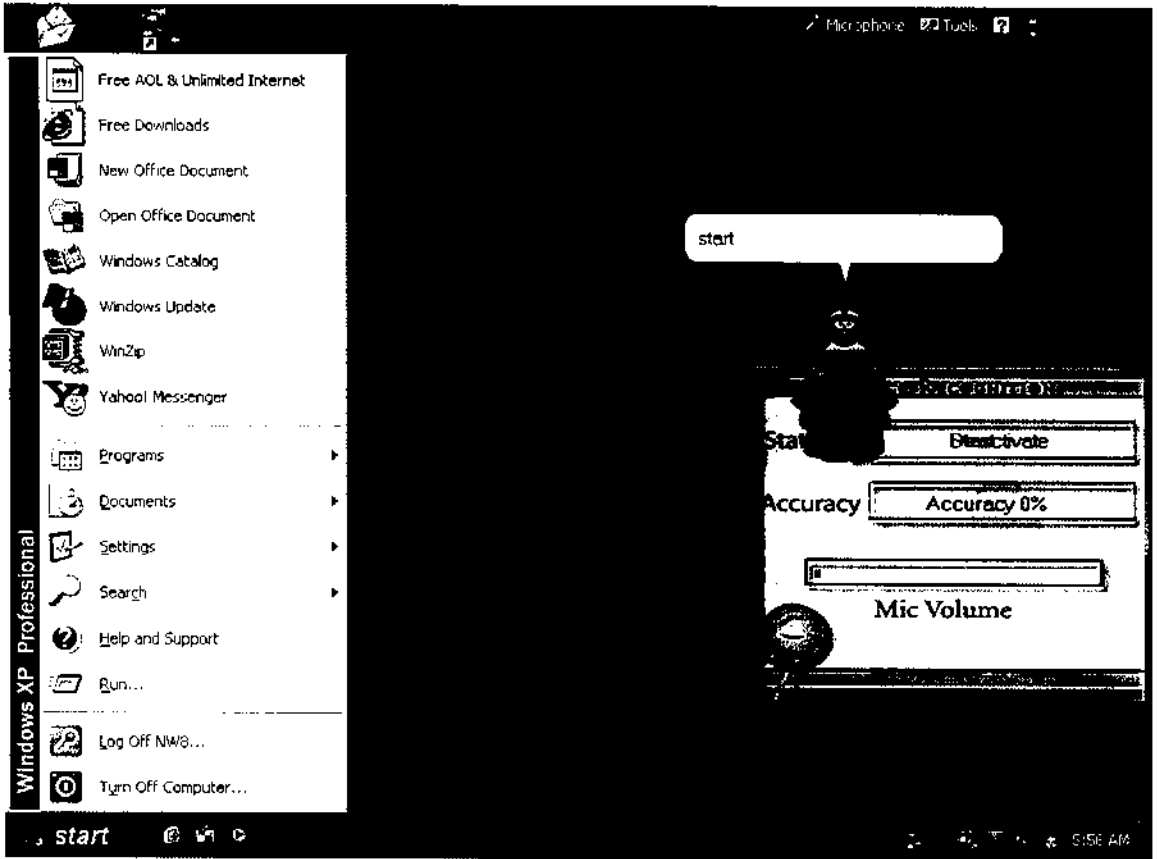


Fig A.6

## *REFERENCES*

---

## 9. REFERENCES

1. "Professional C# &.NET version 1.1", by Robinson and Allen, Wrox publications.
2. "C# and .NET in a nutshell", by Hamilton and Mc Donald.
3. "Developing Windows APIs with Visual C# & .NET", by Microsoft press.
4. "Beginning C# & .NET 2003", by Sahad and Raghunathan, Wrox publications.
5. "C# and .NET Platform", 2<sup>nd</sup> Edition Andrew Troelsen (2003) Springer (India) Private Limited.
6. "MICROSOFT .NET", by David S Platt(2001), Prentice-Hall of India Private Limited.
7. "MICROSOFT VISUAL C# .NET", by John Sharp and John Jagger(2003) ,Prentice-hall of India Private Limited.
8. <http://www.compuiter.org/toc/html>.
9. <http://www.csharp-help.com>
10. <http://www.ai-depot.com>