DEVELOPMENT OF AN ENHANCED SPEECH-TO-TEXT SYSTEM FOR ASSISTING INDIVIDUALS WITH HAND DISABILITIES

BY

AKURE MICHAEL OLUWASEUN
HND/23/COM/FT/0367

A PROJECT REPORT SUBMITTED TO THE

DEPARTMENT OF COMPUTER SCIENCE

INSTITUTE OF INFORMATION AND COMMUNICATION TECHNOLOGY, KWARA STATE POLYTECHNIC ILORIN

IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE AWARD OF HIGHER NATIONAL DIPLOMA (HND) IN COMPUTER SCIENCE

2025

CERTIFICATION

This is to certify that this project is written by Akure Michael oluwaseun, with matriculation number HND/23/COM/FT/0367 in Computer Science Department, Institute of Information and Communication Technology, Kwara State Polytechnic, Ilorin.

-------------------------------------	----------------------
MR. ISIAKA O.S.	Date
(Project Supervisor)	
-------------------------------	---------------------
MR. OYEDEPO F.S.	Date
(Head of Department)	
-----------------------------------	---------------------
External Supervisor	Date

DEDICATION

I dedicate this project to Almighty God for sparing my life throughout my curse of study, also to my lovely parent in person of Akure Janet Sala for her moral and financial supports during my study may Almighty God Blessing you all. Finally, it is dedicated to myself.

AKNOWLEDGEMENT

All praise, adoration glorification to Almighty God who gave me the golden opportunity to be among the existing soul, the provider of all things in life the beginning and end of the life, the creator of my soul, and the wisest in the whole universe, for giving me the strength to complete the pursuit.

I am also grateful to my supervisor Mr. Isiaka O.S for the time he took to supervise the project work, I pray that Almighty God continue to bless you and protect you.

Thanks also to my HOD Mr. Oyedepo F.S, for his guiding and mentorship. May God reward you beautifully.

My profound appreciation goes to my beloved and caring Mother Akure Janet Sala and my siblings for contributing, financially, morally, spiritually to my life, may God protect and guide them and make them reap the fruit of their labour on me.

TABLE OF CONTENT

ABSTRACT

The rapid advancement of speech-to-text technology offers an innovative way for users to input data into systems using their voice, similar to how they would use a keyboard or mouse. This project aims to assist individuals with hand disabilities by developing a speech-to-text software system that enables speech input for computer use. Users will be able to input data via a microphone, with the system converting audio signals into corresponding text strings. The application will support basic operations like saving, opening, and exiting files in a text editor and launching system programs such as MS Paint, Notepad, and Calculator. The software will be developed using JAVA, with potential for future enhancements to support additional functionalities.

CHAPTER ONE

GENERAL INTRODUCTION

1.1 INTRODUCTION

Speech-to-text technology is increasingly becoming a significant tool in making our daily lives more efficient. This software enables users to control their devices and input data by simply speaking, which is particularly beneficial for individuals who have difficulty using their hands. The development of such software not only enhances the efficiency of daily activities but also broadens the scope of accessibility for users with physical disabilities.

Speech-to-text software has become increasingly prominent, with applications now widely integrated into various devices to enhance our daily interactions. For instance, in mobile phone applications, users can simply speak the name of a contact to initiate a call, bypassing the need to type. Similarly, text messaging can be performed by speaking into the phone, eliminating the need for keyboard input. This use of speech for system control not only adds convenience but also reduces production costs in industrial settings.

Speech is a multifaceted form of communication, carrying a wealth of information, including linguistic content, speaker attributes (such as emotional state, regional accent, and physiological characteristics), and environmental context (like the location where the speech occurs). Despite the complexity of this information, humans are adept at interpreting it. This inherent ability has motivated researchers to develop software capable of emulating these interpretive skills. Efforts have been directed towards tasks such as speaker recognition, language detection, speech transcription, translation, and comprehension (Adami, 2003).

Speech-to-text systems function by capturing spoken input through a microphone, which is then converted into text by the computer. This technology not only boosts the efficiency of daily tasks but also enriches the user experience by diversifying interaction methods.

1.2 STATEMENT OF THE PROBLEM

A computer speech-to-text software system is a technological advancement in human-computer interaction. Due to the difficulties encountered by handicapped people, ranging from mild repetitive stress injuries to people with little keyboard skills or experience who are slow typists or do not have the time or resources to develop skills, and also for people who use keyboard a lot and develop Repetitive Strain Injuries, muscular dystrophy, this research on computer speech-to-text software tries to solve these problems.

1.3 AIM AND OBJECTIVES

The aim of this project is to develop software that can accept speech input and convert it to text in order to assist people who have difficulty using their hands. This aim is achieved through the following objectives:

i. Developing software for speech to text conversion.
ii. Designing and developing an interactive user-friendly editor that accepts user speech as input, manipulates speech, and converts speech to text using well-known familiar commands.

1.4 SIGNIFICANCE OF THE STUDY

Speech-to-text software is a branch of Human Computer Interaction that helps people who are physically disabled, i.e. people who can't type with their hands. Earlier work on speech input systems had some constraints in its mode of operation, requiring you to speak in a staccato fashion and leave a gap between two words being input into the system, and any errors must be corrected as soon as they occur. This project research reduces the constraints to the bare minimum, allowing words to be input at a normal pace without leaving distinct pauses between words. However, because the computer is relying solely on your spoken words, words must be entered clearly.

1.5 SCOPE OF THE STUDY

This project involves the designing of software that includes a text editor that converts speech to text. The study's main theme is to allow the user to speak into the microphone that is already connected to the computer, and the text will be displayed on the text editor screen.

1.6 ORGANIZATION OF THE REPORT

The report is divided into five chapters. The first chapter includes an introduction, a statement of the problem, the study's aim and objectives, the significance of the study, the scope of the study, the report's organization, and a glossary of terms. The second chapter discusses a review of related previous work, a review of general studies, an overview of speech-to-text software system, types of speech-to-text software, speech-to-text software system components, their application in various perspectives, and a few speech-to-text software system software. In Chapter Three, the Methodology and Analysis of the Existing System are listed. The fourth chapter goes over the design, implementation, and documentation of the system, as well as the output, input file, and procedure design. The hardware and software support for this project are also discussed. The fifth chapter is the conclusion, and it includes a summary of the entire chapter as well as recommendations.

CHAPTER TWO

LITERATURE REVIEW

2.1 REVIEW OF RELATED WORKS

Su & Hla (2015) implemented a Speech-to-Text conversion system using MFCC for feature extraction and HMM as recognizers. Fifty audio files are recoded and analyzed in the speech database to generate feature vectors. These characteristics are first modeled in the HMM. Following that, the HMM forward algorithm is used to address the test spoken word. Based on the simulation results, it is clear that the average recognition rate of 87.6 percent achieved by the number of states (N=5) is more accurate than any other state. However, if the number of states is too large, there are not enough observations per state to train the model. As a result, the system's performance may suffer. Thus, the number of states in the HMM is an important case in recognition. Using an end point detection algorithm in the preprocessing stage improves the accuracy and reliability of the system in this work.

Kumbharana (2007) established the various objectives for the proposed research project. And he is able to achieve the majority of his research objectives. The researcher created software that performs Gujarati character dictation. To accomplish this goal, he created the software in a Windows environment using VC++ from Visual Studio and Matlab. VC++ is used to create the front end, which interacts with the user for speech input, and Matlab is used to create the algorithms that process the speech signals. After comparing the character to the master database, VC++ invokes the Matlab function for dictation of the pronounced character. Typically, a large number of computer users are accustomed to working in the Microsoft Office environment. Keeping this in mind, the researcher designed and developed a text editor called 'ckksedit' for the front-end purpose, which functions similarly to Microsoft Word.

Taabish (2014) discovered that the two factors that define the performance of speech recognition systems are speed and accuracy. Accuracy can be measured in terms of performance accuracy, which is usually expressed as a word error rate (WER), whereas speed is expressed as a real time factor. The parameter used to calculate the speed of an ASR system is the real time factor (RTF). Language is one of the most important means of communication among humans, and the primary medium used for this is speech. A number of practical limitations have been encountered, which frequently obstructs the widespread deployment of ASR-provided applications and services. The articles provided a brief overview of the state-of-the-art in ASR systems. Through this article, an attempt has been made to provide a review of how much this technology has emerged in the last seven decades as a result of the never-ending efforts of researchers all over the world.

Julien, Lauren, and Fanny (2013) demonstrated that vowel identity is mostly preserved and confirmed consonants' special functional role during lexical identification. While vowels played an important role in the word-detection step that precedes word recognition in difficult listening conditions, consonants appear to be primarily used to identify the lexical item. We also confirmed that CVC is a particularly resistant syllable structure; we found an asymmetry between the ending and beginning of words, with CV onset being less resistant than VC coda. We also identified different perceptual consonant confusion groups based on where they occurred in the words: (b-v, f-p, f-b, m-b s-z p-k-t, k-g m-n) in onsets and (p-b, p-t-k, k-g m-n) in codas. Aside from the acoustic cues of consonants and vowels identified as important for speech recognition, the study revealed that listeners may have access to some acoustic cues of the CV transition that we did not specifically track and that should be more thoroughly investigated in the future in a complementary study using the same type of noise.

According to Karpagavalli (2011), speech is the primary and most convenient mode of communication between people. Building automated systems capable of understanding spoken language and recognizing speech in the same way that humans do is a difficult task. The goal of automatic speech recognition research is to address various speech recognition issues. Various methodologies have been identified and applied to the ASR area, resulting in numerous successful ASR applications in limited domains. Some of the research areas that are gaining traction are robust speech recognition, multimodal speech recognition, and multilingual speech recognition. Many more ASR applications with improved performance are likely in the future.

2.2 AN OVERVIEW OF SPEECH-TO-TEXT SOFTWARE

Speech-to-text software is a technology that allows a computer to capture human speech using a microphone. These words are later recognized by the Speech recognizer, and the system finally outputs the recognized words. The process of speech -to-text software consists of several steps, which will be discussed in detail in the following sections. A Speech-to-text software engine recognizes all words uttered by a human in an ideal situation, but in practice, the performance of a Speech-to-text software engine is dependent on a number of factors. Vocabularies, multiple users, and a noisy environment are the most important factors for a speech-to-text software engine.

2.2.1 TYPES OF SPEECH-TO-TEXT SOFTWARE

Speech-to-text software systems are classified into several classes based on their ability to recognize words and the number of words in their vocabulary. Several types of speech-to-text software are classified as follows:

i. Isolated Speech: Isolated words typically involve a pause between two utterances; this does not imply that it only accepts a single word, but rather that one utterance at a time is required (Rabiner & Juang, 1993).

ii. Connected Speech: Connected words, also known as connected speech, are similar to isolated speech in that they allow for separate utterances with minimal pause between them.

iii. Continuous Speech: Continuous Speech, also known as computer dictation, allows the user to speak almost naturally.

iv. Spontaneous Speech: At its most basic, it can be defined as natural-sounding, unrehearsed speech. An ASR system with spontaneous Speech capability should be able to handle a wide range of natural Speech features such as word runs, "ums" and "ahs," and even minor stutters.

2.3 SPEECH-TO-TEXT SOFTWARE PROCESS

It is a step-by-step process for obtaining text from speech. Speech is input into the system via microphone, the input words are then converted from analog to digital, the acoustic model then creates a statistical representation of the sounds that make up each word and breaks them down into phonemes, the language modeling compares the phonemes to its built-in dictionary, and once the word is recognized, the corresponding text strings are displayed.

Figure 2.1: Speech-to-text software Process

Source: www.slideshare.com

2.3.1 COMPONENTS OF SPEECH-TO-TEXT SOFTWARE

i. Speech Input: When audio is input to the system via a microphone, the computer sound card generates an equivalent digital representation of the received audio.

ii. Digitization: Digitization is the process of converting an analog signal into a digital signal, which includes both the sampling and quantization processes. Sampling is the process of converting a continuous signal to a discrete signal, whereas quantization is the process of approximating a continuous range of values.

iii. Acoustic Model: An acoustic model is created by using software to create statistical representations of the sounds that make up each word from audio recordings of speech and text transcriptions. A speech-to-text software engine uses it to recognize speech. The software acoustic model separates words into phonemes.

iv. Language Model: Language modeling is used in many natural language processing applications, such as speech-to-text software, to capture language properties and predict the next word in a speech sequence. The phonemes are compared to words in the software language model's built-in dictionary.

v. Speech engine: The speech-to-text software engine's job is to convert the input audio into text. To do so, it employs a variety of data, software algorithms, and statistics. Its first operation, as previously discussed, is digitization, which converts it into a suitable format for further processing. Once the audio signal has been properly formatted, it searches for the best match for it. It does this by considering the words it knows; once the signal is recognized, it returns the text string associated with it.

vi. Display: This is the final step in the process, and it displays the corresponding text strings.

2.4 USES OF SPEECH-TO-TEXT SOFTWARE

Essentially, speech-to-text software is used for dictation, which is the translation of spoken words into text, and it can also manipulate and format the text using well-known familiar commands. Writing by speech allows a person to write 150 words per minute or more if he or she can speak that fast. This viewpoint of speech-to-text software programs creates an easy way for composing text and allows people in that industry to compose millions of words digitally in a short period of time rather than writing them one by one, saving time and effort. A keyboard substitute is speech-to-text software. If you are unable to type, speech-to-text software allows you to do almost anything you could do with a keyboard and mouse.

2.5 APPLICATIONS OF SPEECH-TO-TEXT SOFTWARE SYSTEM

i. Medical Perspective: Speech-to-text software programs can help people with disabilities. In such cases, speech-to-text software is especially useful for people who have difficulty using their hands. Speech-to-text software programs are extremely useful and can be used to operate computers. Speech-to-text software, such as speech mail to text, is used in deaf telephony.

ii. Military Perspective: Speech-to-text software programs are important from a military standpoint; in the Air Force, speech-to-text software has the potential to significantly reduce pilot workload. Aside from the Air Force, such programs can be trained for use in helicopters, battle management, and other applications.

iii. Education Perspective: Individuals with learning disabilities who struggle with thought-to-paper communication (basically, they think of an idea but it is processed incorrectly, resulting in a different outcome on paper) can benefit from the software.

iv. Command and Control: Command and Control systems are ASR systems that are designed to perform functions and actions on the system. and iv. Command and Control: Command and Control systems are ASR systems that are designed to perform functions and actions on the system. Utterances such as "Open Netscape" and "Start a new browser" will accomplish this.

v. Telephony: In some Speech Mail systems, callers can speak commands rather than pressing buttons to send specific tones.

vi. Medical/Disabilities: Many people struggle to type due to physical limitations such as repetitive strain injuries (RSI), muscular dystrophy, and others. People who have difficulty hearing, for example, could use a system linked to their phone to convert the caller's speech to text.

2.6 SPEECH-TO-TEXT SOFTWARE WEAKNESS AND FLAWS

Despite all of these advantages and benefits, a perfect Speech-to-text software system cannot be developed. A variety of factors can impair the accuracy and performance of a speech-to-text software program. The speech-to-text software process is simple for humans, but it is a difficult task for machines when compared to the human mind. Speech-to-text software programs appear less intelligent because a human mind is a God-given thing with the ability to think, understand, and react naturally, whereas a computer program must first understand the spoken words in terms of their meanings, and it must create a sufficient balance between the words, noise, and spaces. A human has the ability to filter out noise from speech, whereas a machine requires training and assistance in separating the Speech sound from other sounds.

Few factors that are considerable in this regard are:

i. Homonyms: These are words that are spelled differently and have different meanings but acquire the same meaning, such as "there," "their," "be," and "bee." It is difficult for a computer machine to distinguish between similar-sounding phrases.

ii. Overlapping Speeches: A second difficulty in the process is understanding the Speech uttered by different users; current systems have difficulty distinguishing simultaneous Speeches from multiple users.

iii. Noise factor: the program requires hearing the words spoken by a human clearly and distinctly. Any extra sound can cause interference; first, keep the system away from noisy environments, and then speak clearly; otherwise, the machine will become confused and mix up the words (Ruchi, 2009).

The future of Speech-to-text software.

Dictation speech-to-text software will gradually gain acceptance, and accuracy will improve. Because people frequently misspeak and make unintentional mistakes, "intelligent systems" will be used more frequently to try to guess what the speaker intended to say rather than what was actually said. Microphone and sound systems will be designed to adapt more quickly to changing background noise levels and different environments, as well as to recognize extraneous material to be discarded.

2.7 FEW SPEECH-TO-TEXT SOFTWARE SOFTWARES

i. XSpeech: XSpeech is a dictation/continuous Speech recognizer that can be used in conjunction with a variety of XWindow applications. This software is primarily intended for end users. HomePage:http://www.compapp.dcu.ie/~tdoris/Xspeech/ http://www.zachary.com/creemer/xspeech.html http://xspeech.sourceforge.net/project

ii. ISIP: Mississippi State University's Institute for Signal and Information Processing has made its speech-to-text software engine available. A frontend, a decoder, and a training module are all included in the toolkit. This software is primarily intended for programmers. The toolkit (as well as additional information about ISIP) can be found at: http://www.isip.msstate.edu/project/Speech

iii. Ears: Although not fully developed, Ears is a good starting point for programmers interested in ASR. This software is primarily intended for programmers. FTP:/svrftp.eng.cam.ac.uk/comp.Speech/recognition/

iv. CMU Sphinix: Sphinx was developed at CMU and was recently made open source. This is a fairly large program with a plethora of tools and information. It is still "under construction," but it includes trainers, recognizers, acoustic models, language models, and some limited documentation. This software is primarily intended for programmers. http://www.Speech.cs.cmu.edu/sphinx/Sphinx.html

v. NICO ANN Toolkit: The NICO Artificial Neural Network Toolkit is a versatile back propagation neural network toolkit designed specifically for speech-to-text software applications. This software is primarily intended for programmers. Its website is at http://www.Speech.kth.se/NICO/index.html.

CHAPTER THREE

METHODOLOGY AND ANALYSIS OF EXISTING SYSTEM

3.1 RESEARCH METHODOLOGY

In this study, the Hidden Markov Model (HMM) and Acoustic approach were used to detect and correct linguistic irregularities, as well as to increase robustness for speech recognition in noise.

Hidden Markov Model (HMM)

HMM, as stochastic modeling tools, have the advantage of providing a natural and highly reliable method of speech recognition for a wide range of applications. Because the HMM integrates well into the system in terms of corporating acoustic and syntax information. It is the most widely used method for speech-to-text software at the moment. HMM are "doubly stochastic processes," with the observed data viewed as the result of passing the true (hidden) process through a function that generates the second process (observe). The hidden process is composed of a series of states connected by transitions. Each transition is described by two sets of probabilities: a transition probability, which specifies the likelihood of changing states, and an output probability density function, which specifies the conditional probability of observations.

Acoustic Phonetic Approach

This is the basis for the acoustic phonetic approach, which holds that spoken language has finite, distinct phonetic units that are broadly defined by a set of acoustic properties that manifest in the speech signal over time. The first step in the acoustic phonetic approach is a spectral analysis of speech combined with feature detection, which converts the spectral measurement to a set of features that describe the broad acoustic properties of the various phonetic units. The speech signal is then segmented into stable acoustic regions, and one or more phonetic labels are attached to each segmented region, resulting in a phoneme lattice characterization of the speech. The final step in this method attempts to generate a valid word from the phonetic label sequences generated by segmentation and labeling.

Use case diagram

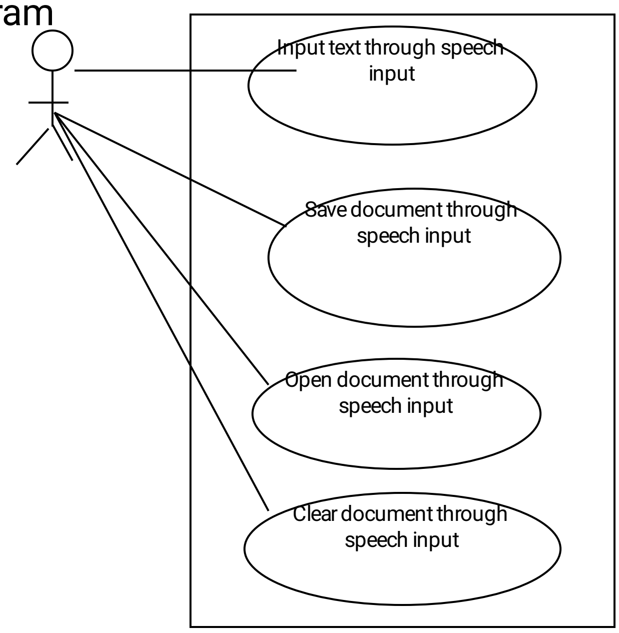

Input text through speech input
Save document through speech input
Open document through speech input
Clear document through speech input

Figure 3.1: Use case diagram

In this use case diagram for the development of speech-to-text software, the user enters text into the system via speech input and then manipulates the text using some known commands via speech.

3.2 ANALYSIS OF THE EXISTING SYSTEM

Speech-to-text software provides a new method of interacting with a computer, and it has great potential as an alternative to using a keyboard and mouse for both people with and without disabilities. The existing system involves the use of a microphone to provide input into the system, which then converts your spoken words into text to be displayed on a text editor. The existing system forces you to speak in a staccato manner, which wastes time.

3.3 PROBLEM OF THE EXISTING SYSTEM

The existing system is plagued by the following issues:

i. Noise: The existing system is disturbed by noise in the background where words are being entered into the system.

ii. Slower speed: The existing system is slow.

iii. Accuracy: The existing system was not very accurate, as the words displayed by the text editor did not correspond to the words entered.

iv. Time Consumption: The existing system was discovered to be slow and time consuming due to the fact that it allows you to speak in a staccato fashion, requiring you to leave a gap between two words being spoken.

3.4 DESCRIPTION OF THE PROPOSED SYSTEM

The new system addresses the problems of existing system theory by reducing it to its most basic form. The following are the main characteristics of the proposed system.

a. The new system will be able to accept speech input through a microphone and display it on the computer screen.

b. The new system will create an interactive text editor that will allow the user to enter text, manipulate text, and format text using well-known and familiar commands.

c. The new system will be able to launch system applications such as MS Paint, Notepad, and Calculator.

3.5 ADVANTAGES OF THE PROPOSED SYSTEM

i. The system is simple to use.

ii. It is precise and quick.

iii. It boosts efficiency.

iv. It significantly reduces wrist and finger wear and tear.

V. It broadens one's horizons.

CHAPTER FOUR

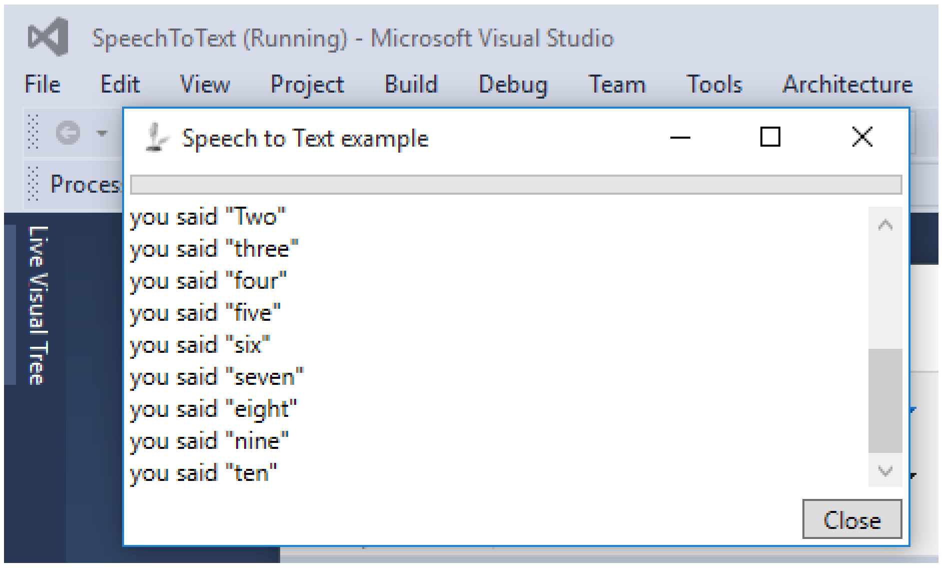
DESIGN AND IMPLEMENTATION OF THE SYSTEM

4.1 DESIGN OF THE SYSTEM

This is the process of designing the input, output, and processing steps to meet the user's needs as identified in the system analysis. It begins with logical design, which results in a specification of the system's major features. Data, input, output, processing, storage, and other control requirements are among them.

4.1.1 OUTPUT DESIGN

The output is referred to as a report, processing result, message, and so on, and is directly related to input files that are stored and processed, resulting in output.



```
you said "Two"
you said "three"
you said "four"
you said "five"
you said "six"
you said "seven"
you said "eight"
you said "nine"
you said "ten"
```

Figure 4.1: Screenshot of an interface that recognizes the pronunciation of numbers one through ten.

4.1.2 INPUT DESIGN

The system's input referred to the speech or words spoken that the system was to recognize.

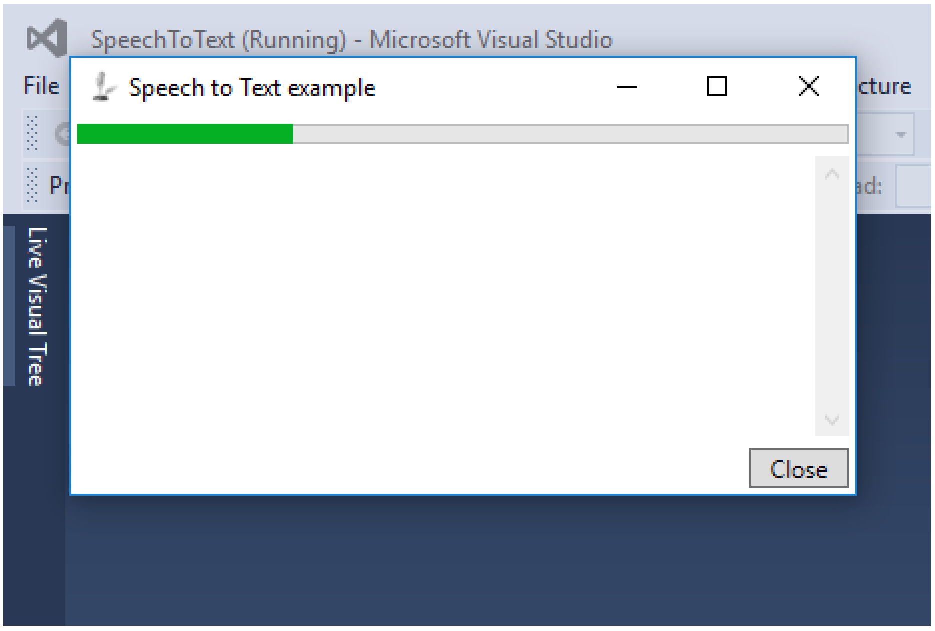
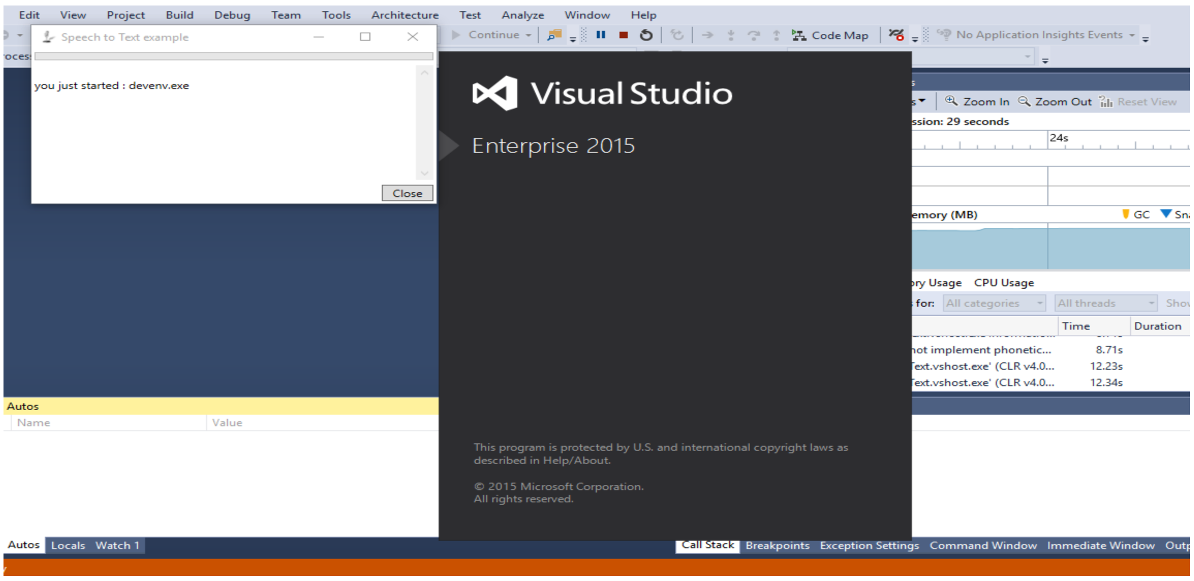


Figure 4.2: Screenshot of an interface that is awaiting speech-to-text software

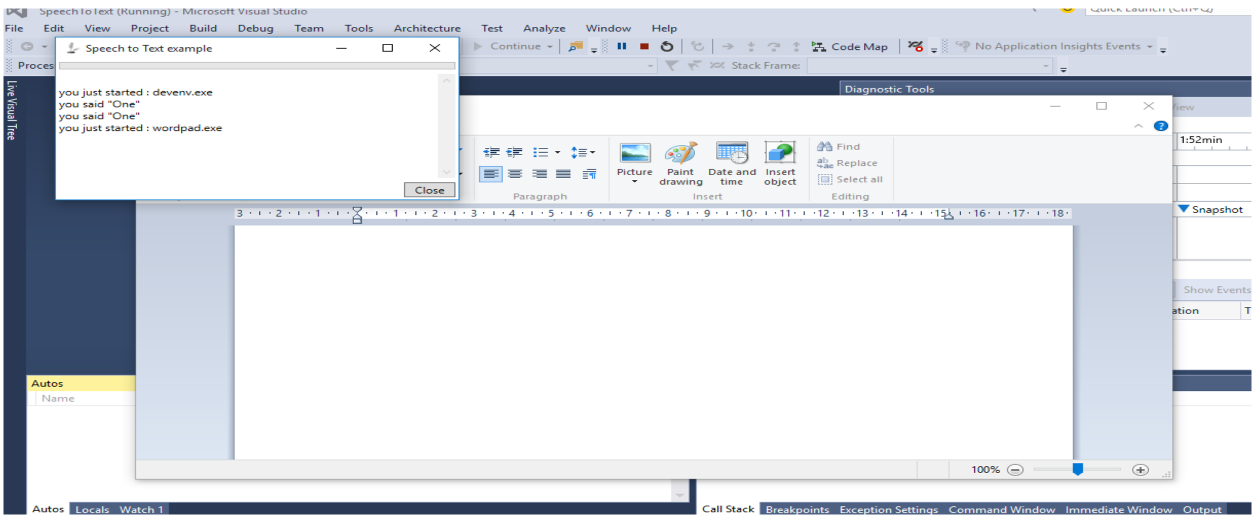

Edit View Project Build Debug Team Tools Architecture Test Analyze Window Help
Speech to Text example
Close
Continue
Code Map
No Application Insights Events
Visual Studio
Enterprise 2015
Zoom In
Zoom Out
Reset View
Autos
Name
Value
Autos Locals Watch 1
Call Stack Breakpoints Exception Settings Command Window Immediate Window

```
you just started : devenv.exe
```

This program is protected by U.S. and international copyright laws as described in Help/About.

© 2015 Microsoft Corporation.
All rights reserved.

Figure 4.3: Screenshot of the interface that recognizes and launches the Visual Studio program


SpeechToText (Running) - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Architecture Test Analyze Window Help
Speech to Text example
Close
Code Map
Diagnostic Tools
Picture
Paint drawing
Date and time
Insert object
Find
Replace
Select all
Paragraph
Insert
Editing
100%
Autos Locals Watch 1
Call Stack Breakpoints Exception Settings Command Window Immediate Window Output

```
you just started : devenv.exe
you said "One"
you said "One"
you just started : wordpad.exe
```

Figure 4.4: Screenshot of the interface that recognizes and launches the Wordpad application

4.1.3 DATABASE DESIGN

A database is an integrated collection of data that represents entities important to the operation of an individual or organization; it is organized to reflect the logical relationship between data elements. A data base is at the heart of any information management system. Two distinct data base structures were used in the design. This section provides detailed information on all data used in the design of this project; it provides analysis on the type, length, and so on.

4.1.4 PROCEDURE DESIGN

The arrangement, steps, and flow of a program are referred to as procedure design. It is concerned with how the commands and instructions are grouped together to form the entire program. The program's methodology is an event-driven programming language, which makes it more efficient in terms of execution, development time, storage utilization, and maintenance.

4.2 IMPLEMENTATION OF THE SYSTEM

This section is concerned with the new system's implementation and documentation. It is concerned with the physical requirements of the machines on which it can run, such as memory capacity and operating system. It also includes a comprehensive user's manual or guide, as well as various screens (forms) as they appear to the user during run-time.

4.2.1 CHOICE OF PROGRAMMING

JAVA is used as the programming language, and MYSQL is used as the database backend. Various languages were examined during the analysis of the programming language to be used, but JAVA was chosen because it has the excellent capability to learn speech patterns and this engine can handle any non-supported situation gracefully, and it is also architectural neutral. Observation and internet research are used to gather information.

4.2.2 HARDWARE REQUIREMENT

In order for the system to function properly, the following minimum hardware configuration is required:

i. 2.0 GHz Intel Pentium IV Processor or higher
ii. 1GB of RAM (4GB recommended)
iii. A minimum of 256GB of available disk space
iv. Input devices (keyboard and mouse)

4.2.3 SOFTWARE REQUIREMENT

This system's software requirements are as follows:

i. Operating System: Minimum of Windows 7, Windows 8, 8.1, Windows 10,
ii. Linux, Apple MAC,
iii. Wamp Server for Windows,
iv. Mozilla Firefox, Netscape Navigator, Internet Explorer,
v. Opera Browser, and Google Chrome are examples of web browsers.

4.2.4 IMPLEMENTATION TECHNIQUES

The changeover technique used for this project's implementation is a parallel technique in which the new and old systems run concurrently.

4.3 DOCUMENTATION OF THE SYSTEM

4.3.1 PROGRAM DOCUMENTATION

Working with software requires proper and adequate documentation, which is part of the quality of a good program. You click the start button, then locate all programs and click on Speech-to-text software home screen, enter your password and username, and the speech-to-text software page will appear, listing all available friends in green for online friends and red for offline friends. As a result, you can select any menu from the list provided in order to perform any operation you desire.

4.3.2 OPERATING THE SYSTEM

To operate the system, the following algorithm must be followed:-

i. Switch the System on
ii. Allow it to boot and load all the startup data and variables
iii. Click on start menu
iv. select the package name and wait for the application to load.

The user can navigate and carry out the operation of the menu item by selecting one from the list of menus that appears on the application's main page.

4.3.3 MAINTAINING THE SYSTEM

The system's built-in safeguards prevent illegal or unintentional data file modification. Provisions for safely storing completed transactions, balances, and statements and restoring this information if necessary, and the software's ability to accurately recover from an accidental or improper shutdown. The system's features are well-suited to the needs of the social network. Integration refers to how well the system's various components communicate with one another, allowing data sharing and reducing the need for multiple data entries (the need to input the same data into different parts of the system).

CHAPTER FIVE

SUMMARY, CONCLUSION AND RECOMMENDATION

5.1 SUMMARY OF FINDINGS

This project explored the development of speech-to-text software to aid individuals with hand disabilities, enabling them to interact with computers using their voice. By converting speech into text, this software provides an alternative to traditional input methods like keyboards and mice. The system comprises several components, including speech input, digitization, acoustic modeling, language modeling, and the speech engine, which work together to deliver accurate text representation of spoken words. The software also includes an interactive text editor and the ability to launch system programs, enhancing its utility. Various methodologies, such as the Hidden Markov Model (HMM) and Acoustic Phonetic Approach, were employed to improve the software's accuracy and performance. Despite some challenges like noise interference and the need for clear speech, the project successfully demonstrated the potential of speech-to-text technology in improving accessibility and efficiency for users with disabilities.

5.2 CONCLUSION

The development of speech-to-text software marks a significant advancement in human-computer interaction, particularly for individuals with hand disabilities. This technology allows users to input data through speech, converting spoken words into digital text efficiently and accurately. The project aimed to create user-friendly software that not only translates speech into text but also performs basic commands and launches various applications. The use of JAVA programming language facilitated the creation of a robust speech-to-text application that is both accessible and effective.

5.3 RECOMMENDATION

To further enhance the functionality and usability of speech-to-text software, future research and development should focus on the following areas:

i. Noise Reduction: Implement advanced noise-cancellation techniques to improve the accuracy of speech recognition in noisy environments.

ii. Speed Optimization: Enhance the processing speed to reduce latency and improve the real-time responsiveness of the software.

iii. Customization: Allow users to customize commands and vocabulary to better suit their individual needs and preferences.

iv. Multilingual Support: Expand the software's capabilities to support multiple languages, making it accessible to a wider audience.

v. Integration with More Applications: Increase the range of system programs and third-party applications that the software can interact with, providing greater versatility.

REFERENCES

Charu J. (2008). Speech Recognition. Retrieved from the internet. Source: http://www.scribd.com/doc/2586608/Speechrecognition.pdf

John K. (2003). Speech recognition technologies. The Emergence of Human Computer Interaction and Speech Recognition System.

John K. (2003). Speech Recognition Technologies. Retrieved from the internet. Source: http://electronics.howstuffworks.com/gadgets/high-tech-gadgets/speechrecognition.htm/printables

Juang B. H. & Lawrence R. R. (2004). Automatic Speech Recognition – A Brief History of the Technology Development. Retrieved from the internet. Source: http://www.ece.ucsb.edu/Faculty/Rabiner/ece259/Reprints/354_LALI-ASRHistory-final-10-8.pdf

Julien K., Lauren M. & Fanny J. (2013). Speech Recognition in Natural Background Noise.

Jurafsky D. & Martin J. (2000). Speech and Language Processing: An Introduction to Natural Language Processing. Computational Linguistics and Speech Recognition ISBN: 0130950696.

Karpagavalli S. (2011). Automatic Speech Recognition.

Ksenia S. (2007). Automatic Speech Recognition System. Retrieved from the internet. Source:http://www.cs.bris.ac.uk/Teaching/Resources/COMS12303/lectures/Ksenia_Shalonova-Speech_Recognition.pdf

Kumbharana C. (2007). Speech Pattern Recognition for speech to Text Conversion.

Rabiner L. and Juang B. (1993). Fundamentals of Speech Recognition System. A modern Approach to Speech Recognition System. ISBN: 0130151572.

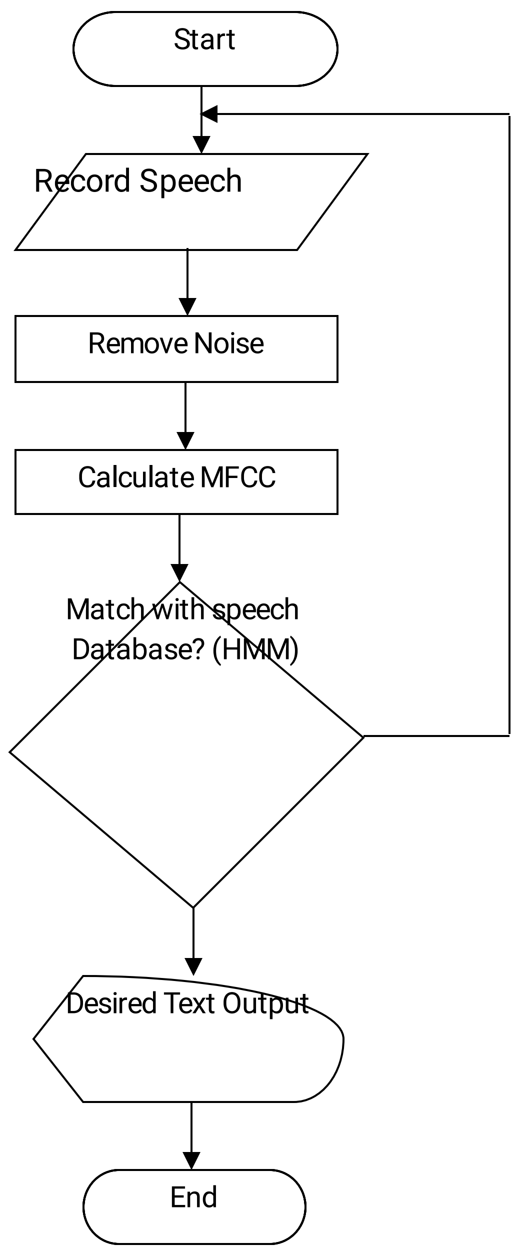
Rajveer K. (2001). Speech recognition- The next revolution of speech Recognition System and What we can do about it, 5th edition.

Stephen C. (2002). Speech Recognition HOWTO. Retrieved from the internet. Source: http://www.abilityhub.com/speech.description.htm

Su M. M. & Hla M. T. (2015). Speech-To-Text Conversion (STT) System Using Hidden Markov Model (HMM).

Taabish G. (2014). A Systematic Analysis of Automatic Speech Recognition.

FLOWCHART


Start
Record Speech
Remove Noise
Calculate MFCC
Match with speech Database? (HMM)
Desired Text Output
End

SOURCE CODE LISTINGS

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Windows;

namespace SpeechToText
{
  /// <summary>
  /// Interaktionslogik für "App.xaml"
  /// </summary>
  public partial class App : Application
  {
  }
}

#region using directives

using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Speech.Recognition;
using System.Windows;
using System.Diagnostics;

#endregion

namespace SpeechToText
{
  /// <summary>
  /// MainWindow class
  /// </summary>
  public partial class MainWindow : Window
  {
    #region locals

    /// <summary>
    /// the engine
    /// </summary>
    SpeechRecognitionEngine speechRecognitionEngine = null;

    /// <summary>
    /// list of predefined commands
    /// </summary>
    List<Word> words = new List<Word>();

    #endregion

    #region ctor

    /// <summary>
    /// Initializes a new instance of the <see cref="MainWindow"/> class.
    /// </summary>
```

```
        public MainWindow()
        {
            InitializeComponent();

            try
            {
                // create the engine
                speechRecognitionEngine = createSpeechEngine("de-DE");

                // hook to events
                speechRecognitionEngine.AudioLevelUpdated += new
EventHandler<AudioLevelUpdatedEventArgs>(engine_AudioLevelUpdated);
                speechRecognitionEngine.SpeechRecognized += new
EventHandler<SpeechRecognizedEventArgs>(engine_SpeechRecognized);

                // load dictionary
                loadGrammarAndCommands();

                // use the system's default microphone
                speechRecognitionEngine.SetInputToDefaultAudioDevice();

                // start listening
                speechRecognitionEngine.RecognizeAsync(RecognizeMode.Multiple);
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "Speech recognition failed");
            }
        }

        #endregion

        #region internal functions and methods

        /// <summary>
        /// Creates the speech engine.
        /// </summary>
        /// <param name="preferredCulture">The preferred culture.</param>
        /// <returns></returns>
        private SpeechRecognitionEngine createSpeechEngine(string preferredCulture)
        {
            foreach (RecognizerInfo config in SpeechRecognitionEngine.InstalledRecognizers())
            {
                if (config.Culture.ToString() == preferredCulture)
                {
                    speechRecognitionEngine = new SpeechRecognitionEngine(config);
                    break;
                }
            }

            // if the desired culture is not found, then load default
            if (speechRecognitionEngine == null)
            {
                MessageBox.Show("The desired culture is not installed on this machine, the speech-engine will
continue using "
                    + SpeechRecognitionEngine.InstalledRecognizers()[0].Culture.ToString() + " as the default
culture.",
                    "Culture " + preferredCulture + " not found!");
                speechRecognitionEngine = new
SpeechRecognitionEngine(SpeechRecognitionEngine.InstalledRecognizers()[0]);
            }

            return speechRecognitionEngine;
        }

        /// <summary>
        /// Loads the grammar and commands.
        /// </summary>
        private void loadGrammarAndCommands()
        {
            try
            {
                Choices texts = new Choices();
                string[] lines = File.ReadAllLines(Environment.CurrentDirectory + "\\example.txt");
                foreach (string line in lines)
                {
                    // skip commentblocks and empty lines..
                    if (line.StartsWith("--") || line == String.Empty) continue;

                    // split the line
                    var parts = line.Split(new char[] { '|' });

                    // add commandItem to the list for later lookup or execution
                    words.Add(new Word() { Text = parts[0], AttachedText = parts[1], IsShellCommand = (parts[2] ==
"true") });

                    // add the text to the known choices of speechengine
                    texts.Add(parts[0]);
                }
                Grammar wordsList = new Grammar(new GrammarBuilder(texts));
                speechRecognitionEngine.LoadGrammar(wordsList);
            }
            catch (Exception ex)
            {
                throw ex;
            }
        }

        /// <summary>
        /// Gets the known command.
        /// </summary>
        /// <param name="command">The order.</param>
        /// <returns></returns>
        private string getKnownTextOrExecute(string command)
        {
            try
            {
                var cmd = words.Where(c => c.Text == command).First();

                if (cmd.IsShellCommand)
                {
                    Process proc = new Process();
                    proc.EnableRaisingEvents = false;
                    proc.StartInfo.FileName = cmd.AttachedText;
                    proc.Start();
                    return "you just started : " + cmd.AttachedText;
                }
                else
                {
                    return cmd.AttachedText;
                }
            }
            catch (Exception)
            {
                return command;
            }
        }

        #endregion

        #region speechEngine events

        /// <summary>
        /// Handles the SpeechRecognized event of the engine control.
        /// </summary>
        /// <param name="sender">The source of the event.</param>
        /// <param name="e">The <see cref="System.Speech.Recognition.SpeechRecognizedEventArgs"/>
instance containing the event data.</param>
        void engine_SpeechRecognized(object sender, SpeechRecognizedEventArgs e)
        {
            txtSpoken.Text += "\r" + getKnownTextOrExecute(e.Result.Text);
            scvText.ScrollToEnd();
        }

        /// <summary>
        /// Handles the AudioLevelUpdated event of the engine control.
        /// </summary>
        /// <param name="sender">The source of the event.</param>
        /// <param name="e">The <see cref="System.Speech.Recognition.AudioLevelUpdatedEventArgs"/>
instance containing the event data.</param>
        void engine_AudioLevelUpdated(object sender, AudioLevelUpdatedEventArgs e)
        {
            prgLevel.Value = e.AudioLevel;
        }

        #endregion

        #region window closing

        /// <summary>
        /// Handles the Closing event of the Window control.
        /// </summary>
        /// <param name="sender">The source of the event.</param>
        /// <param name="e">The <see cref="System.ComponentModel.CancelEventArgs"/> instance
containing the event data.</param>
        private void Window_Closing(object sender, System.ComponentModel.CancelEventArgs e)
        {
            // unhook events
            speechRecognitionEngine.RecognizeAsyncStop();
            // clean references
            speechRecognitionEngine.Dispose();
        }

        #endregion
```

```
        #region GUI events

        private void Button_Click(object sender, RoutedEventArgs e)
        {
            this.Close();
        }

        #endregion
    }
}\

<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="4.0" DefaultTargets="Build" xmlns="http://schemas.microsoft.com/developer/
msbuild/2003">
  <PropertyGroup>
    <Configuration Condition=" '$(Configuration)' == '' ">Debug</Configuration>
    <Platform Condition=" '$(Platform)' == '' ">x86</Platform>
    <ProductVersion>8.0.30703</ProductVersion>
    <SchemaVersion>2.0</SchemaVersion>
    <ProjectGuid>{344D6A98-4DF4-499E-8B4F-DF2D10738A7F}</ProjectGuid>
    <OutputType>WinExe</OutputType>
    <AppDesignerFolder>Properties</AppDesignerFolder>
    <RootNamespace>SpeechToText</RootNamespace>
    <AssemblyName>SpeechToText</AssemblyName>
    <TargetFrameworkVersion>v4.0</TargetFrameworkVersion>
    <TargetFrameworkProfile>Client</TargetFrameworkProfile>
    <FileAlignment>512</FileAlignment>
    <ProjectTypeGuids>{60dc8134-eba5-43b8-bcc9-bb4bc16c2548};{FAE04EC0-301F-11D3-
BF4B-00C04F79EFBC}</ProjectTypeGuids>
    <WarningLevel>4</WarningLevel>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Debug|x86' ">
    <PlatformTarget>x86</PlatformTarget>
    <DebugSymbols>true</DebugSymbols>
    <DebugType>full</DebugType>
    <Optimize>false</Optimize>
    <OutputPath>bin\Debug\</OutputPath>
    <DefineConstants>DEBUG;TRACE</DefineConstants>
    <ErrorReport>prompt</ErrorReport>
    <WarningLevel>4</WarningLevel>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Release|x86' ">
    <PlatformTarget>x86</PlatformTarget>
    <DebugType>pdbonly</DebugType>
    <Optimize>true</Optimize>
    <OutputPath>bin\Release\</OutputPath>
    <DefineConstants>TRACE</DefineConstants>
    <ErrorReport>prompt</ErrorReport>
    <WarningLevel>4</WarningLevel>
  </PropertyGroup>
  <PropertyGroup>
    <ApplicationIcon>Images\speech.ico</ApplicationIcon>
  </PropertyGroup>
  <PropertyGroup>
    <StartupObject>
    </StartupObject>
  </PropertyGroup>
  <ItemGroup>
    <Reference Include="System" />
    <Reference Include="System.Data" />
    <Reference Include="System.Speech" />
    <Reference Include="System.Xml" />
    <Reference Include="Microsoft.CSharp" />
    <Reference Include="System.Core" />
    <Reference Include="System.Xml.Linq" />
    <Reference Include="System.Data.DataSetExtensions" />
    <Reference Include="System.Xaml">
      <RequiredTargetFramework>4.0</RequiredTargetFramework>
    </Reference>
    <Reference Include="WindowsBase" />
    <Reference Include="PresentationCore" />
    <Reference Include="PresentationFramework" />
  </ItemGroup>
  <ItemGroup>
    <ApplicationDefinition Include="App.xaml">
      <Generator>MSBuild:Compile</Generator>
      <SubType>Designer</SubType>
    </ApplicationDefinition>
    <Page Include="MainWindow.xaml">
      <Generator>MSBuild:Compile</Generator>
      <SubType>Designer</SubType>
    </Page>
    <Compile Include="App.xaml.cs">
      <DependentUpon>App.xaml</DependentUpon>
      <SubType>Code</SubType>
    </Compile>
    <Compile Include="MainWindow.xaml.cs">
      <DependentUpon>MainWindow.xaml</DependentUpon>
      <SubType>Code</SubType>
    </Compile>
  </ItemGroup>
  <ItemGroup>
    <Compile Include="Properties\AssemblyInfo.cs" />
    <Compile Include="Word.cs" />
    <AppDesigner Include="Properties\" />
  </ItemGroup>
  <ItemGroup>
    <Resource Include="Images\speech.ico" />
  </ItemGroup>
  <ItemGroup>
    <EmbeddedResource Include="example.txt">
      <CopyToOutputDirectory>Always</CopyToOutputDirectory>
    </EmbeddedResource>
  </ItemGroup>
  <Import Project="$(MSBuildToolsPath)\Microsoft.CSharp.targets" />
  <!-- To modify your build process, add your task inside one of the targets below and uncomment it.
       Other similar extension points exist, see Microsoft.Common.targets.
  <Target Name="BeforeBuild">
  </Target>
  <Target Name="AfterBuild">
  </Target>
  -->
</Project>
```