#### **SPEECH TO TEXT CONVERSION**

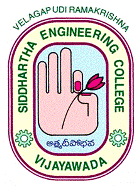
Project report submitted in partial fulfillment of the Requirements for the Award of the Degree of **B.Tech** in **Computer Science and Engineering**

By

**NIKITA K. KHATRI (Y09CS022)**

**K. SINDHU (Y09CS026)**

**K. SRUTHI (Y09CS030)**



Under the Esteemed Guidance of

**Dr. V. SRINIVASA RAO, M.Tech, Ph.D**

**Professor and Head of the Department**

**Department of Computer Science and Engineering**

**V.R.Siddhartha Engineering College**

**(Autonomous)**

**(Affiliated to Acharya Nagarjuna University)**

**Approved by AICTE- Accredited by NBA**

**Vijayawada-520007**

**2012-2013**

**SPEECH TO TEXT CONVERSION**

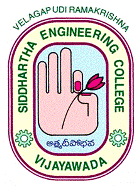
Project report submitted in partial fulfillment of the requirement for the award of the Degree of **B.Tech** in **Computer Sciences and Engineering**

By

**NIKITA K. KHATRI (Y09CS022)**

**K. SINDHU (Y09CS026)**

**K. SRUTHI (Y09CS030)**

****

Under the Esteemed Guidance of

**Dr. V. SRINIVASA RAO, M.Tech, Ph.D**

**Professor and Head of the Department**

**Department of Computer Science and Engineering**

**V.R.Siddhartha Engineering College**

**(Autonomous)**

**(Affiliated to Acharya Nagarjuna University)**

**Approved by AICTE- Accredited by NBA**

**Vijayawada-520007**

**2012-2013**

###### **CERTIFICATE**

This is to certify that the project report entitled SPEECH TO TEXT CONVERSION being submitted by

NIKITA.K.KHATRI Y09CS022

K.SINDHU Y09CS026

K.SRUTHI Y09CS030

in partial fulfillment for the award of the Degree of Bachelor of Technology in Computer Science and Engineering to the Acharya Nagarjuna University is a record of bonafied work carried out under my guidance and supervision.

The results embodied in this Mini project report have not been submitted to any other University or Institute for the award of any Degree or Diploma.

(Dr. V. Srinivasa Rao)

M.Tech, Ph.D, HOD Head of the Department

**ACKNOWLEDGMENT**

Our most sincere and graceful acknowledgement is due to its sanctum **Velagapudi Ramakrishna Siddhartha Engineering** College for giving this opportunity to fill our aspirations and to become engineers.

We take the opportunity to express our deep sense of gratitude and gracefulness to our project guide **Dr. V. SRINIVASA RAO, M.Tech, Ph.D**, Head of the Department for his excellent guidance, constant encouragement and support during all phases of work.

We wish to express our profound thanks to **Dr. V. SRINIVASA RAO, M.Tech, Ph.D**, HOD of C.S.E for providing necessary facilities to make this project success.

We wish to express our profound thanks to **Dr. G. SAMBASIVA RAO,** principal of V.R.Siddharatha Engineering College, for providing necessary facilities to make this project success.

We extend our sincere and whole hearted thanks to all those who helped us completing this project successfully.

**ABSTRACT**

Most people use oral language for everyday communication, i.e. they speak to other people and hear what other people say. If people who are severely impaired in their hearing abilities want to take part in oral communication, they need a way to compensate their physical impairment. Hearing aids are sufficient for many hearing impairment people. However, if hearing aids are insufficient, spoken language has to be transferred into a modality which is accessible without hearing. i.e. into the visual domain.

Speech-to-text-conversion is a useful tool for integrating people with hearing impairments in oral communication settings, e. g. counseling interviews or conferences. This application helps user to give the speech as input and as output it gets the written language which is 100% correct and understandable. This application contains speech acquisition where speech samples are obtained from the speaker in real time and stored in memory for pre-processing, speech to text conversion where each sample of speech is converted into text and displayed to end user. Speech to text conversion is applied in health care, military, telephony and other domains, and further applications include aerospace (e.g. [space exploration](http://en.wikipedia.org/wiki/Space_exploration), [spacecraft](http://en.wikipedia.org/wiki/Spacecraft), etc.), [Court reporting](http://en.wikipedia.org/wiki/Court_reporting) (Realtime Speech Writing), robotics, telematics (e.g., vehicle Navigation Systems) etc.

**CONTENTS**

1. Introduction
   1. Scope of the Project
   2. Vision of the Project
2. Requirement Analysis
   1. Problem Definition
   2. Existing System
   3. Proposed System

2.3.1) Benefits

2..3.2)Applications

1. System Requirement

3.1) Software Specification

3.2) Hardware Specification

4. Software Design

4.1) Design

4.2) Introduction to Unified Modeling Language

4.2.1) Behavioral Diagrams

4.2.2) Structural Diagrams

4.3) UML Diagrams

4.3.1) Use case Diagram

4.3.2) Class Diagram

4.3.3) Sequence Diagram

4.3.4) Collaboration Diagram

4.3.5) State Diagram

4.3.6) Activity Diagram

5. Software Overview

5.1) .net Framework

5.2) WPF

5.3) C# .net

6. Implementation and Coding

6.1) WPF coding

6.2) .net coding

7. Testing

7.1) Introduction to Testing

7.2) Testing Strategies

7.3) Test cases

8. Screen Shots

9. Conclusion

10. Further Enhancements/Recommendation

11. Bibliography

**1. Introduction**

* 1. **Scope of the Project**

Speech-to-text-conversion is a useful tool for integrating people with hearing impairments in oral communication. Hearing aids are sufficient for many hearing impairment people. However, if hearing aids are insufficient, spoken language has to be transferred into a modality which is accessible without hearing i.e., into the visual domain. This application helps user to give the speech as input and as output it gets the written language.

* 1. **Vision of the Project**

To give people access to spoken words and auditory events almost simultaneously with the realization of the original sound event. STT is fast enough in producing written language. We get a word-by-word transfer enhanced by a description of auditory events from the surroundings as well as adaptations of the original wording into easier forms of language. The written words are presented in a way that is optimally recognizable and understandable for the readers.

**2. Requirement Analysis**

**2.1. Problem Definition**

Hearing aids are sufficient for many hearing impairment people. However, if hearing aids are insufficient, spoken language has to be transferred into a modality which is accessible without hearing i.e., into the visual domain.

**2.2. Existing System**

There are a number of systems that have been proposed world-wide for Speech-to-Text System. Digital recognition machine for a single user was built. Vowel region of each digit was focused. Pattern recognition ideas were used. Clustering algorithm were used to determine the number of distinct patterns.

**2.3. Proposed System**

Speech-to-text-conversion is a useful tool for integrating people with hearing impairments in oral communication. Due to shift in technology, Hidden Markov Model (HMM) is used to find the most probabilistic result for the input speech.HMM uses Hidden Markov Tool Kit (HTK)

**2.3.1. Benefits**

* Fast enough in producing written language.
* Word-by-word transfer enhanced by a description of auditory events from the surroundings as well as adaptations of the original wording into easier forms of language.
* The written words are presented in a way that is optimally recognizable and understandable for the readers.

**2.3.2. Applications**

* Health care
* Military
* Telephony
* Aerospace (e.g., space exploration, spacecraft etc.)
* Court reporting (Real time Speech Writing)
* Robotics
* Telematics (e.g., vehicle Navigation Systems) etc.

**3. System Requirement**

**3.1. Software Specification**

* Operating system : Windows XP and above
* Coding Language : C#.Net
* Other technology : WPF

**3.2. Hardware Specification**

* System : Intel Pentium IV and above
* Hard Disk : 40 GB
* Ram : 256 Mb
* HD microphone

**4. Software Design**

**4.1. Design**

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer’s goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement have been specified and analyzed, system design is the first of the three technical activities – design, code and test that is required to build and verify software.

The importance can be stated with a single word “Quality” design is the place where quality is fostered in software development. Design provides us with representations of software that can assess for quality. Design is the only way that we can accurately translate a customer’s view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage.

During design, progressive refinement of date structure, program structure, and procedural details are developed reviewed and documented. System design can be viewed from either technical or project management perspective. From the technical point of view, design is comprised of four activities – architectural design, data structure design, interface design and procedural design.

**4.2. Introduction to Unified Modeling Language (UML)**

The Unified Modeling Language (UML) is a language for specifying, constructing, visualizing, and documenting the artifacts of a software-intensive system. Analogous to the use of architectural blueprints in the construction industry, UML provides a common language for describing software models, and it can be used in conjunction with a wide range of software lifecycles and development processes.

**4.2.1. Behavioral Diagrams**

**1. USE CASE DIAGRAM**

Use Case diagrams identify the functionality provided by the system (use cases), the users who interact with the system (actors), and the association between the users and the functionality. Use Cases are used in the Analysis phase of software development to articulate the high-level requirements of the system.

The primary goals of Use Case diagrams include:

* Providing a high-level view of what the system does
* Identifying the users ("actors") of the system
* Determining areas needing human-computer interfaces.

Use Cases extend beyond pictorial diagrams. In fact, text-based use case descriptions are often used to supplement diagrams, and explore use case functionality in more detail.

1. **SEQUENCE DIAGRAM**

Sequence diagrams document the interactions between classes to achieve a result, such as a use case. Because UML is designed for object-oriented programming, these communications between classes are known as messages. The Sequence diagram lists objects horizontally, and time vertically, and models these messages over time.

Basic Sequence Diagram Symbols and Notations:

In a Sequence diagram, classes and actors are listed as columns, with vertical lifelines indicating the lifetime of the object over time.

**Object:** Objects are instances of classes, and are arranged horizontally. The pictorial representation for an Object is a class (a rectangle) with the name prefixed by the object name (optional) and a semi-colon.

**Actor**: Actors can also communicate with objects, so they too can be listed as a column. An Actor is modeled using the ubiquitous symbol, the stick figure.

**Lifeline:** The Lifeline identifies the existence of the object over time. The notation for a Lifeline is a vertical dotted line extending from an object.

**Activation**: Activations, modeled as rectangular boxes on the lifeline, indicate when the object is performing an action.

**Message**: Messages, modeled as horizontal arrows between Activations, indicate the communications between objects.

**Destroying Objects**: Objects can be terminated early using an arrow labeled "< < destroy > >" that points to an X.

**Loops**: A repetition or loop within a sequence diagram is depicted as a rectangle. Place the condition for exiting the loop at the bottom left corner in square brackets [  ].

1. **ACTIVITY DIAGRAM**

Activity diagrams are used to document workflows in a system, from the business level down to the operational level. When looking at an Activity diagram, you'll notice elements from State diagrams. In fact, the Activity diagram is a variation of the state diagram where the "states" represent operations, and the transitions represent the activities that happen when the operation is complete. The general purpose of Activity diagrams is to focus on flows driven by internal processing vs. external events.

Basic Activity Diagram Symbols and Notations

**Activity States**: Activity states mark an action by an object. The notations for these states are rounded rectangles, the same notation as found in State chart diagrams.

**Transition**: When an Activity State is completed, processing moves to another Activity State. Transitions are used to mark this movement. Transitions are modeled using arrows.

**Swim lane**: Swim lanes divide activities according to objects by arranging objects in column format and placing activities by that object within that column. Objects are listed at the top of the column, and vertical bars separate the columns to form the swim lanes.

**Initial State**: The Initial State marks the entry point and the initial Activity State. The notation for the Initial State is the same as in State chart diagrams, a solid circle. There can only be one Initial State on a diagram.

**Final State**: Final States mark the end of the modeled workflow. There can be multiple Final States on a diagram, and these states are modeled using a solid circle surrounded by another circle.

**Synchronization Bar**: Activities often can be done in parallel. To split processing ("fork”) or to resume processing when multiple activities have been completed ("join"), Synchronization Bars are used. These are modeled as solid rectangles, with multiple transitions going in and/or out.

**Branching**: A diamond represents a decision with alternate paths. The outgoing alternates should be labeled with a condition or guard expression. You can also label one of the paths "else."

**4.2.2. Structural Diagrams**

1. **CLASS DIAGRAM**

Class diagram are used to describe the type of objects and their relationship by providing a static, structural view of a system. Class diagrams show the classes of the system, their interrelationships (including inheritance, aggregation, and association), and the operations and attributes of the classes. Class diagrams are used for a wide variety of purposes, including both conceptual/domain modeling and detailed design modeling.

1. **OBJECT DIAGRAM**

Object diagram model instances of classes. This type of diagram is used to describe the system at a particular point in time. Using this technique, we can validate the class diagram and its multiplicity rules with real-world data, and record test scenarios. From a notation standpoint, object diagrams borrow elements from class diagrams.

1. **COMPONENT DIAGRAM**

Component diagrams fall under the category of an implementation diagram, a kind of diagram that models the implementation and deployment of the system. A Component Diagram, in particular, is used to describe the dependencies between various software components such as the dependency between executable files and source files. This information is similar to that within make files, which describe source code dependencies and can be used to properly compile an application.

**4.3. UML Diagrams**

**4.3.1. Use case Diagram**



**4.3.2. Class Diagram**



**4.3.3. Sequence Diagram**



**4.3.4. Collaboration Diagram**



**4.3.5. State Diagram**



**4.3.6. Activity Diagram**



**5. Software Overview**

**5.1. JAVA TECHNOLOGY**

Java technology is both a programming language and a platform. It is a general purpose, concurrent, class-based, object oriented language. Java is currently one of the most popular programming languages in use.

**The Java Programming Language**

The java programming language is a high-level language that can be characterized by all the following buzzwords:

* Platform Independent
* Simple
* Object Oriented
* Robust
* Distributed
* Portable
* Dynamic
* Secure
* Performance
* Multithreaded

**Platform Independent**

The concept of Write-once-run-anywhere (known as the Platform independent) is one of the important key feature of java language that makes java as the most powerful language. Not even a single language is idle to this feature but java is more closer to this feature. The programs written on one platform can run on any platform provided the platform must have the JVM.

**Simple**

There are various features that make the java as a simple language. Programs are easy to write and debug because java does not use the pointers explicitly. It is much harder to write the java programs that can crash the system but we cannot say about the other programming languages. Java provides the bug free system due to the strong memory management. It also has the automatic memory allocation and deallocation system.

**Object Oriented**

To be an Object Oriented language, any language must follow at least the four characteristics.

* Inheritance: It is the process of creating the new classes and using the behavior of the existing classes by extending them just to reuse the existing code and adding the additional features as needed.
* Encapsulation: It is the mechanism of combining the information and providing the abstraction.
* Polymorphism: As the name suggest one name multiple form, Polymorphism is the way of providing the different functionality by the   
     functions having the same name based on the signatures of the methods.
* Dynamic binding: Sometimes we don't have the knowledge of objects about their specific types while writing our code. It is the way of providing the maximum functionality to a program about the specific type at runtime.

As the languages like Objective C, C++ fulfills the above four characteristics yet they are not fully object oriented languages because they are structured as well as object oriented languages. But in case of java, it is a fully Object Oriented language because object is at the outer most level of data structure in java. No stand alone methods, constants, and variables are there in java. Everything in java is object even the primitive data types can also be converted into object by using the wrapper class.

**Robust**

Java has the strong memory allocation and automatic garbage collection mechanism. It provides the powerful exception handling and type checking mechanism as compare to other programming languages. Compiler checks the program whether there any error and interpreter checks any run time error and makes the system secure from crash. All of the above features make the java language robust.

**Distributed**

The widely used protocols like HTTP and FTP are developed in java. Internet programmers can call functions on these protocols and can get access the files from any remote machine on the internet rather than writing codes on their local system.

**Portable**

The feature Write-once-run-anywhere makes the java language portable provided that the system must have interpreter for the JVM. Java also have the standard data size irrespective of operating system or the processor. These features make the java as a portable language.

**Dynamic**

While executing the java program the user can get the required files dynamically from a local drive or from a computer thousands of miles away from the user just by connecting with the Internet.

**Secure**

Java does not use memory pointers explicitly. All the programs in java are run under an area known as the sand box. Security manager determines the accessibility options of a class like reading and writing a file to the local disk. Java uses the public key encryption system to allow the java applications to transmit over the internet in the secure encrypted form. The bytecode Verifier checks the classes after loading.

**Performance**

Java uses native code usage, and lightweight process called threads. In the beginning interpretation of bytecode resulted the performance slow but the advance version of JVM uses the adaptive and just in time compilation technique that improves the performance.

**Multithreaded**

As we all know several features of Java like Secure, Robust, Portable, dynamic etc; you will be more delighted to know another feature of Java which is **Multithreaded.**Java is also a Multithreaded programming language. Multithreading means a single program having different threads executing independently at the same time. Multiple threads execute instructions according to the program code in a process or a program. Multithreading works the similar way as multiple processes run on one computer.    
Multithreading programming is a very interesting concept in Java. In multithreaded programs not even a single thread disturbs the execution of other thread. Threads are obtained from the pool of available ready to run threads and they run on the system CPUs. This is how Multithreading works in Java which you will soon come to know in details in later chapters.

**Interpreted**

We all know that Java is an interpreted language as well. With an interpreted language such as Java, programs run directly from the source code.   
The interpreter program reads the source code and translates it on the fly into computations. Thus, Java as an interpreted language depends on an interpreter program.  
The versatility of being **platform independent** makes Java to outshine from other languages. The source code to be written and distributed is platform independent.    
Another advantage of Java as an interpreted language is its error debugging quality. Due to this any error occurring in the program gets traced. This is how it is different to work with Java.

**Architecture Neutral**

The term architectural neutral seems to be weird, but yes Java is an architectural neutral language as well. The growing popularity of networks makes developers think distributed. In the world of network it is essential that the applications must be able to migrate easily to different computer systems. Not only to computer systems but to a wide variety of hardware architecture and Operating system architectures as well.  The Java compiler does this by generating byte code instructions, to be easily interpreted on any machine and to be easily translated into native machine code on the fly.The compiler generates an architecture-neutral object file format to enable a Java application to execute anywhere on the network and then the compiled code is executed on many processors, given the presence of the Java runtime system.Hence Java was designed to support applications on network. This feature of Java has thrived the programming language.

In the Java programming language, all source code is first written in plain text files ending with the .java extension. Those source files are then compiled into .class files by the javac compiler. A .class file does not contain code that is native to your processor; it instead contains bytecodes — the machine language of the Java Virtual Machine (Java VM). The java launcher tool then runs your application with an instance of the Java Virtual Machine.



**An overview of the software development process**

Because the Java VM is available on many different operating systems, the same .class files are capable of running on Microsoft Windows, the Solaris™ Operating System (Solaris OS), Linux, or Mac OS. Some virtual machines, such as the Java SE Hot Spot at a Glance, perform additional steps at runtime to give your application a performance boost. This includes various tasks such as finding performance bottlenecks and recompiling (to native code) frequently used sections of code.



Through the Java VM, the same application is capable of running on multiple platforms.

## The Java Platform

## A platform is the hardware or software environment in which a program runs. We've already mentioned some of the most popular platforms like Microsoft Windows, Linux, Solaris OS, and Mac OS. Most platforms can be described as a combination of the operating system and underlying hardware. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other hardware-based platforms.

The Java platform has two components:

* The Java Virtual Machine
* The Java Application Programming Interface (API)

You've already been introduced to the Java Virtual Machine; it's the base for the Java platform and is ported onto various hardware-based platforms.

The API is a large collection of ready-made software components that provide many useful capabilities. It is grouped into libraries of related classes and interfaces; these libraries are known as packages. The next section What Can Java Technology Do? highlights some of the functionality provided by the API.



The API and Java Virtual Machine insulate the program from the underlying hardware.

As a platform-independent environment, the Java platform can be a bit slower than native code. However, advances in compiler and virtual machine technologies are bringing performance close to that of native code without threatening portability.

The terms "Java Virtual Machine" and "JVM" mean a Virtual Machine for the Java platform.

5.2. **XML**

Main features of XML:

1. XML files are text files, which can be managed by any text editor.
2. XML is very simple, because it has less than 10 syntax rules.
3. XML is extensible, because it only specifies the structural rules of tags. No specification on tags them self.

Because of these features, XML offers following **advantages**

1. XML provides a basic syntax that can be used to share information between different kinds of computers, different applications, and different organizations. XML data is stored in plain text format. This software- and hardware-independent way of storing data allows different incompatible systems to share data without needing to pass them through many layers of conversion. This also makes it easier to expand or upgrade to new operating systems, new applications, or new browsers, without losing any data.
2. With XML, your data can be available to all kinds of "reading machines" (Handheld computers, voice machines, news feeds, etc), and make it more available for blind people, or people with other disabilities.
3. XML provides a gateway for communication between applications, even applications on wildly different systems. As long as applications can share data (through HTTP, file sharing, or another mechanism), and have an XML parser, they can share structured information that is easily processed. Databases can trade tables, business applications can trade updates, and document systems can share information.
4. It supports Unicode, allowing almost any information in any written human language to be communicated.
5. It can represent common computer science data structures: records, lists and trees.
6. Its self-documenting format describes structure and field names as well as specific values.
7. The strict syntax and parsing requirements make the necessary parsing algorithms extremely simple, efficient, and consistent.
8. Content-based XML markup enhances searchability, making it possible for agents and search engines to categorize data instead of wasting processing power on context-based full-text searches.
9. XML is heavily used as a format for document storage and processing, both online and offline.
10. It is based on international standards.
11. It can be updated incrementally.
12. It allows validation using schema languages such as XSD and Schematron, which makes effective unit-testing, firewalls, acceptance testing, contractual specification and software construction easier.
13. The hierarchical structure is suitable for most (but not all) types of documents.
14. It is platform-independent, thus relatively immune to changes in technology.
15. Forward and backward compatibility are relatively easy to maintain despite changes in DTD or Schema.
16. Its predecessor, SGML, has been in use since 1986, so there is extensive experience and software available.

**5.3. Android**

Android is a Linux-based open source operating system primarily for touch screen mobile devices such as smart phones and tablet computers. It delivers a complete set of software for mobile devices: an operating system, middleware and key mobile applications.

**5.1.1 Features**

**Handset layouts**

The platform is adaptable to larger and traditional smart phone layouts.

**Storage**

SQLite, a lightweight relational database, is used for data storage purposes.

**Connectivity**

Android supports connectivity technologies including GSM/EDGE, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, NFC and WiMAX.

**Messaging**

SMS and MMS are available forms of messaging, including threaded text messaging and Android Cloud To Device Messaging (C2DM) and now enhanced version of C2DM, Android Google Cloud Messaging (GCM) is also a part of Android Push Messaging service.

**Multiple language support**

Android supports multiple languages.

**Web browser**

The web browser available in Android is based on the open-source WebKit layout engine, coupled with Chrome's V8 JavaScript engine. The browser scores 100/100 on the Acid3 test on Android 4.0.

**Java support**

While most Android applications are written in Java, there is no Java Virtual Machine in the platform and Java byte code is not executed. Java classes are compiled into Dalvik executables and run on Dalvik, a specialized virtual machine designed specifically for Android and optimized for battery-powered mobile devices with limited memory and CPU. J2ME support can be provided via third-party applications.

**Media support**

Android supports the following audio/video/still media formats: WebM, H.263, H.264 (in 3GP or MP4 container), MPEG-4 SP, AMR, AMR-WB (in 3GP container), AAC, HE-AAC(in MP4 or 3GP container), MP3, MIDI, Ogg Vorbis,  FLAC,  WAV,  JPEG,  PNG, GIF, BMP, WebP.

**Streaming media support**

Adobe Flash Streaming (RTMP) and HTTP Dynamic streaming are supported by the Flash plug-in. Apple HTTP Live Streaming is supported by RealPlayer for Android, and by the operating system in Android 3.0 (Honeycomb).

**Additional hardware support**

Android can use video/still cameras,  touchscreens,  GPS,  accelerometers,  gyroscopes,  barometers, magnetometers, dedicated gaming controls, proximity and pressure sensors, thermometers, accelerated 2D bit blits (with hardware orientation, scaling, pixel format conversion) and accelerated 3D graphics.

**Multi-touch**

Android has native support for multi-touch which was initially made available in handsets such as the HTC Hero. The feature was originally disabled at the kernel level (possibly to avoid infringing Apple's patents on touch-screen technology at the time). Google has since released an update for the Nexus One and the Motorola Droid which enables multi-touch natively.

**Bluetooth**

Supports A2DP, AVRCP, sending files (OPP), accessing the phone book (PBAP), voice dialing and sending contacts between phones. Keyboard, mouse and joystick (HID) support is available in Android 3.1+, and in earlier versions through manufacturer customizations and third-party applications.

**Video calling**

Android does not support native video calling, but some handsets have a customized version of the operating system that supports it, either via the UMTS network (like the Samsung Galaxy S) or over IP. Video calling through Google Talk is available in Android 2.3.4 and later. Gingerbread allows Nexus S to place Internet calls with a SIP account. This allows for enhanced VoIP dialing to other SIP accounts and even phone numbers. Skype 2.1 offers video calling in Android 2.3, including front camera support. Users with the Google+ android app can video chat with other google+ users through hangouts.

**Multitasking**

Multitasking of applications, with unique handling of memory allocation, is available.

**Voice based features**

Google search through voice has been available since initial release. Voice actions for calling, texting, navigation, etc. are supported on Android 2.2 onwards. As of Android 4.1, Google has expanded Voice Actions with the ability to talk back and read answers from Google's Knowledge Graph when queried with specific commands. The ability to control hardware has not yet been implemented.

**Tethering**

Android supports tethering, which allows a phone to be used as a wireless/wired Wi-Fi hotspot. Before Android 2.2 this was supported by third-party applications or manufacturer customizations.

**Screen capture**

Android supports capturing a screenshot by pressing the power and volume-down buttons at the same time. Prior to Android 4.0, the only methods of capturing a screenshot were through manufacturer and third-party customizations or otherwise by using a PC connection (DDMS developer's tool). These alternative methods are still available with the latest Android.

**External storage**

Most Android devices include microSD slot and can read microSD cards formatted with FAT32, Ext3 or Ext4 file system. To allow use of high-capacity storage media such as USB flash drives and USB HDDs, many Android tablets also include USB 'A' receptacle. Storage formatted with FAT32 is handled by Linux Kernel VFAT driver, while 3rd party solutions are required to handle other popular file systems such as NTFS, HFS Plus and exFAT.

**6. Implementation and Coding**

**6.1 WPF Coding**

****

**WPF code for this screenshot is**

<Window x:Class="My\_Blog.Login" xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Title="Login" Height="300" Width="500">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="114\*" />

<ColumnDefinition Width="364\*" />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="14\*" />

<RowDefinition Height="338\*" />

</Grid.RowDefinitions>

<Grid.Background>

<LinearGradientBrushEndPoint="1,0.5" StartPoint="0,0.5">

<GradientStop Color="#9900C1FF" Offset="1" />

<GradientStop Color="#FFE6FFFF" Offset="1" />

<GradientStop Color="#6100C1FF" Offset="0" />

<GradientStop Color="#2A00C1FF" Offset="0.291" />

</LinearGradientBrush>

</Grid.Background>

<Canvas Grid.Row="1" Height="251" HorizontalAlignment="Left" Margin="10,0,0,0" Name="canvas1" VerticalAlignment="Top" Width="456" Grid.ColumnSpan="2">

<TextBlockCanvas.Left="172" Canvas.Top="14" Height="23" Name="txtblock\_title" Text="Sign in Form" FontFamily="Verdana" FontSize="20" />

<Label Canvas.Left="111" Canvas.Top="76" Content="User Name " Height="28" Name="lbl\_UserName" FontFamily="Verdana" FontSize="15" />

<Label Canvas.Left="111" Canvas.Top="125" Content="Password" Height="28" Name="lbl\_Password" FontFamily="Verdana" FontSize="15" Width="86" />

<TextBoxCanvas.Left="241" Canvas.Top="80" Height="23" Name="txt\_UserName" Width="120" />

<PasswordBoxCanvas.Left="241" Canvas.Top="125" Height="23" Name="pwdbx\_Password" Width="120" />

<!--<Button Canvas.Left="249" Canvas.Top="190" Content="Cancel" Height="32" Name="btn\_Cancel" Width="75" />-->

<Border BorderThickness="0" BorderBrush="Blue" Width="100" Height="80" RenderTransformOrigin=".5,.5" Margin="19,15,359,166" Canvas.Left="234" Canvas.Top="152">

<Border.RenderTransform>

<ScaleTransform x:Name="scale"/>

</Border.RenderTransform>

<Border.Triggers>

<EventTriggerRoutedEvent="Border.MouseEnter">

<BeginStoryboard>

<Storyboard Duration="0:0:.5" Storyboard.TargetName="scale">

<DoubleAnimationStoryboard.TargetProperty="ScaleX" To="2"/>

<DoubleAnimationStoryboard.TargetProperty="ScaleY" To="2"/>

</Storyboard>

</BeginStoryboard>

</EventTrigger>

<EventTriggerRoutedEvent="Border.MouseLeave">

<BeginStoryboard>

<Storyboard Duration="0:0:.5" Storyboard.TargetName="scale">

<DoubleAnimationStoryboard.TargetProperty="ScaleX" To="1"/>

<DoubleAnimationStoryboard.TargetProperty="ScaleY" To="1"/>

</Storyboard>

</BeginStoryboard>

</EventTrigger>

</Border.Triggers>

<!--<Image Name="image1" Source="/myWpfApplication1;component/Images/user.png" ImageFailed="image1\_ImageFailed" />-->

<Button Canvas.Left="249" Canvas.Top="190" Content="Cancel" Height="32" Name="btn\_Cancel" Width="75" Click="btn\_Cancel\_Click" />

</Border>

<Image Canvas.Left="367" Canvas.Top="6" Height="97" Name="image1" Stretch="Fill" Width="75" Source="H:\sp\Speech to Text\Speech to Text\Speech to Text\Images\images (1).jpg" OpacityMask="#FF83A0CC" Opacity="1" />

<Border BorderBrush="Blue" BorderThickness="0" Height="80" RenderTransformOrigin=".5,.5" Width="100" Margin="133,14,245,167" Canvas.Left="7" Canvas.Top="153">

<Border.RenderTransform>

<ScaleTransform x:Name="scale1" />

</Border.RenderTransform>

<Border.Triggers>

<EventTriggerRoutedEvent="Border.MouseEnter">

<BeginStoryboard>

<BeginStoryboard.Storyboard>

<Storyboard Duration="0:0:.5" Storyboard.TargetName="scale1">

<DoubleAnimationStoryboard.TargetProperty="ScaleX" To="2" />

<DoubleAnimationStoryboard.TargetProperty="ScaleY" To="2" />

</Storyboard>

</BeginStoryboard.Storyboard>

</BeginStoryboard>

</EventTrigger>

<EventTriggerRoutedEvent="Border.MouseLeave">

<BeginStoryboard>

<BeginStoryboard.Storyboard>

<Storyboard Duration="0:0:.5" Storyboard.TargetName="scale1">

<DoubleAnimationStoryboard.TargetProperty="ScaleX" To="1" />

<DoubleAnimationStoryboard.TargetProperty="ScaleY" To="1" />

</Storyboard>

</BeginStoryboard.Storyboard>

</BeginStoryboard>

</EventTrigger>

</Border.Triggers>

<Button Canvas.Left="158" Canvas.Top="191" Content="Sign In" Height="32" Name="btn\_SignIn" Width="75" Click="btn\_SignIn\_Click" />

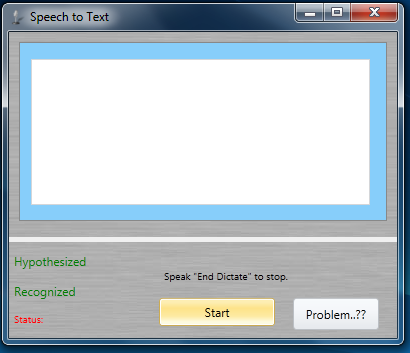
</Border>

</Canvas>

</Grid>

</Window>

**6.2 C# .net coding**



**C# .net code for this screenshot is**

using System;

usingSystem.Collections.Generic;

usingSystem.Linq;

usingSystem.Text;

usingSystem.Windows;

usingSystem.Windows.Controls;

usingSystem.Windows.Data;

usingSystem.Windows.Documents;

usingSystem.Windows.Input;

usingSystem.Windows.Media;

usingSystem.Windows.Media.Imaging;

usingSystem.Windows.Navigation;

usingSystem.Windows.Shapes;

usingSystem.Speech.Recognition;

usingSystem.Threading;

usingSystem.Speech.Synthesis;

namespaceSpeech\_to\_Text

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

//Enum to hold the recognizer's state

privateenum State

{

Idle = 0,

Accepting = 1,

Off = 2,

}

private State RecogState = State.Off;

privateSpeechRecognitionEngine recognizer;

privateSpeechSynthesizer synthesizer = null;

privateint Hypothesized = 0;

privateint Recognized = 0;

publicMainWindow()

{

InitializeComponent();

}

private void Window\_Loaded(object sender, RoutedEventArgs e)

{

//initialize recognizer and synthesizer

InitializeRecognizerSynthesizer();

//if input device found then proceed

if (SelectInputDevice())

{

LoadDictationGrammar();

ButtonStart.IsEnabled = true;

ReadAloud("Speech Engine Ready for Input");

}

}

/// <summary>

/// initialize recognizer and synthesizer along with their events

/// /// </summary>

private void InitializeRecognizerSynthesizer()

{

varselectedRecognizer = (from e in SpeechRecognitionEngine.InstalledRecognizers()

wheree.Culture.Equals(Thread.CurrentThread.CurrentCulture)

select e).FirstOrDefault();

recognizer = new SpeechRecognitionEngine(selectedRecognizer);

recognizer.AudioStateChanged+=new EventHandler<AudioStateChangedEventArgs>(recognizer\_AudioStateChanged);

recognizer.SpeechHypothesized += new EventHandler<SpeechHypothesizedEventArgs>(recognizer\_SpeechHypothesized);

recognizer.SpeechRecognized += new EventHandler<SpeechRecognizedEventArgs>(recognizer\_SpeechRecognized);

synthesizer = new SpeechSynthesizer();

}

/// <summary>

/// select input device if available

/// </summary>

/// <returns></returns>

privateboolSelectInputDevice()

{

boolproceedLoading = true;

//if OS is above XP

if (IsOscompatible())

{

try

{

recognizer.SetInputToDefaultAudioDevice();

}

catch

{

proceedLoading = false; //no audio input device

}

}

//if OS is XP or below

else

ThreadPool.QueueUserWorkItem(InitSpeechRecogniser);

returnproceedLoading;

}

/// <summary>

/// Findout if OS is compatible.

/// </summary>

/// <returns>true if greater than XP otherwise false</returns>

privateboolIsOscompatible()

{

OperatingSystemosInfo = Environment.OSVersion;

if (osInfo.Version> new Version("6.0"))

return true;

else

return false;

}

private void InitSpeechRecogniser(object o)

{

recognizer.SetInputToDefaultAudioDevice();

}

/// <summary>

/// Load grammars, one for command and other for dictation

/// </summary>

private void LoadDictationGrammar()

{

GrammarBuildergrammarBuilder = new GrammarBuilder();

grammarBuilder.Append(new Choices("End Dictate"));

Grammar commandGrammar = new Grammar(grammarBuilder);

commandGrammar.Name = "main command grammar";

recognizer.LoadGrammar(commandGrammar);

DictationGrammardictationGrammar = new DictationGrammar();

dictationGrammar.Name = "dictation";

recognizer.LoadGrammar(dictationGrammar);

}

#region Recognizer events

private void recognizer\_AudioStateChanged(object sender, AudioStateChangedEventArgs e)

{

switch (e.AudioState)

{

caseAudioState.Speech:

LabelStatus.Content = "Listening";

break;

caseAudioState.Silence:

LabelStatus.Content = "Idle";

break;

caseAudioState.Stopped:

LabelStatus.Content = "Stopped";

break;

}

}

private void recognizer\_SpeechHypothesized(object sender, SpeechHypothesizedEventArgs e)

{

Hypothesized++;

LabelHypothesized.Content = "Hypothesized: " + Hypothesized.ToString();

}

private void recognizer\_SpeechRecognized(object sender, SpeechRecognizedEventArgs e)

{

Recognized++;

LabelRecognized.Content = "Recognized: " + Recognized.ToString();

if (RecogState == State.Off)

return;

float accuracy = (float)e.Result.Confidence;

string phrase = e.Result.Text;

{

if (phrase == "End Dictate")

{

RecogState = State.Off;

recognizer.RecognizeAsyncStop();

ReadAloud("Dictation Ended");

return;

}

TextBox1.AppendText(" " + e.Result.Text);

}

}

#endregion

/// <summary>

/// pause recognition and speak the text sent

/// </summary>

/// <param name="speakText"></param>

public void ReadAloud(string speakText)

{

try

{

recognizer.RecognizeAsyncCancel();

synthesizer.SpeakAsync(speakText);

}

catch { }

}

private void ButtonStart\_Click(object sender, RoutedEventArgs e)

{

switch (RecogState)

{

caseState.Off:

RecogState = State.Accepting;

ButtonStart.Content = "Stop";

recognizer.RecognizeAsync(RecognizeMode.Multiple);

break;

caseState.Accepting:

RecogState = State.Off;

ButtonStart.Content = "Start";

recognizer.RecognizeAsyncStop();

break;

}

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

Window1 w = new Window1();

w.Show();

this.Hide();

}

private void Window\_Closing(object sender, System.ComponentModel.CancelEventArgs e)

{

}

}

}

**7. Testing**

**7.1 Introduction to Testing**

Testing is a process, which reveals errors in the program. It is the major quality measure employed during software development. During testing, the program is executed with a set of test cases and the output of the program for the test cases is evaluated to determine if the program is performing as it is expected to perform.

**7.2 TESTING IN STRATEGIES**

In order to make sure that the system does not have errors, the different levels of testing strategies that are applied at differing phases of software development are:

**Unit Testing**

Unit Testing is done on individual modules as they are completed and become executable. It is confined only to the designer's requirements.

**Each module can be tested using the following two Strategies:**

**Black Box Testing**

In this strategy some test cases are generated as input conditions that fully execute all functional requirements for the program. This testing has been uses to find errors in the following categories:

* Incorrect or missing functions
* Interface errors
* Errors in data structure or external database access
* Performance errors
* Initialization and termination errors.

In this testing only the output is checked for correctness. The logical flow of the data is not checked.

**White Box testing**

In this the test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all the cases. It has been uses to generate the test cases in the following cases:

* Guarantee that all independent paths have been executed.
* Execute all logical decisions on their true and false Sides.
* Execute all loops at their boundaries and within their operational bounds
* Execute internal data structures to ensure their validity.

**Integrating Testing**

Integration testing ensures that software and subsystems work together a whole. It tests the interface of all the modules to make sure that the modules behave properly when integrated together.

**System Testing**

Involves in-house testing of the entire system before delivery to the user. It's aim is to satisfy the user the system meets all requirements of the client's specifications.

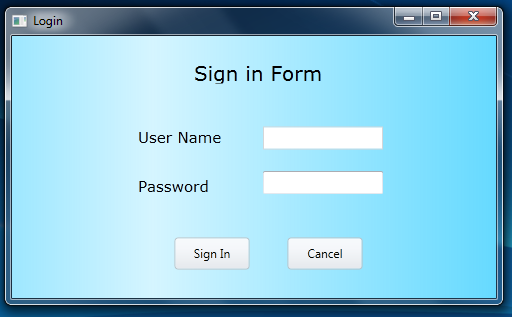
**Acceptance Testing**

It is a pre-delivery testing in which entire system is tested at client's site on real world data to find errors.

**7.3 Test Cases:**

**8. Screenshots**

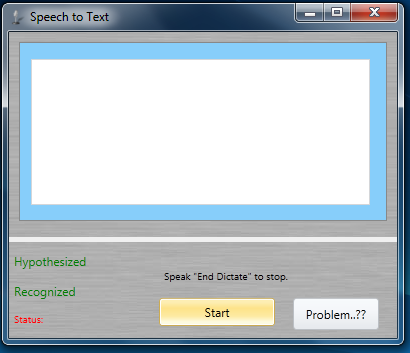
**8.1. User interface**

****

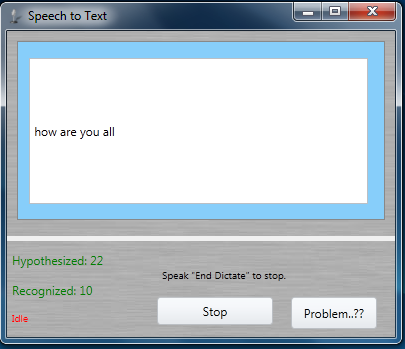
**8.2. User login with name and password**

****

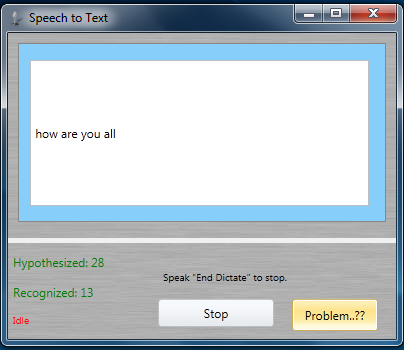
**8.3. Recognising speech**

****

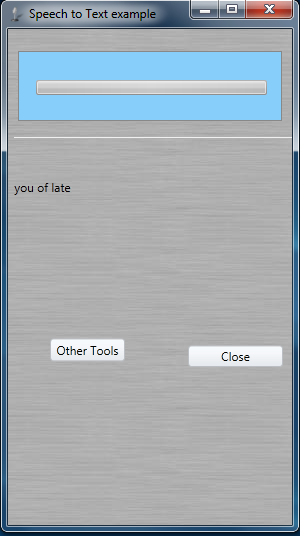
**8.4. Displaying text**

****

**8.5. For any problems**

****

**8.6. Problem module**

****

**9. Conclusion**

Tourism has become the most preferred activity for relaxation or recreation. But, moving to any place which we desire needs a lot of research regarding finding the locations, moving to preferred destinations and also finding accommodation .So we are developing this tourist eye, a smart offline travel guide which provides all such facilities in the palm of user’s hand acting as trip planner and trip journal. Overcoming the disadvantages of the existing system we have provided the user with 100% offline, Explore, Portability, User friendly and Transportation map facilities.