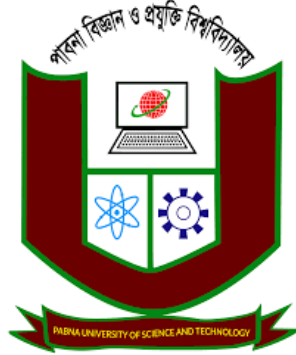


# **Design and Implementation of Bangla Voice Keyboard for Writing Bijoy (ASCII), Unicode (UTF-8) and English Fonts**



**Faculty of Engineering and Technology  
Department of Information and Communication Engineering  
B.Sc. (Engineering) Examination 2020  
Course Code: ICE-4210  
Course Title: Thesis**

**A thesis paper is submitted to the Department of Information and Communication Engineering (ICE), Pabna University of Science and Technology (PUST) in the partial fulfillment of the requirement for the degree of Bachelor of Science in Engineering**

**Submitted by:  
ID: 170626  
Registration No: 1065268  
Session: 2016-17**

**Department of Information and Communication Engineering  
Pabna University of Science and Technology**

---

**Department of Information and Communication Engineering  
Pabna University of Science and Technology  
Pabna-6600, Bangladesh**

**November, 2022**

## **Abstract**

This era of information operated mostly by computer systems requires user input frequently. Most of the inputs are taken in the form of text mostly in the English language. Apart from inputs for operating computer systems, we also tend to express ourselves in social media through our writings. As English is an international language, it is widely used. In the usual scenario, most of the users are comfortable writing in English in the usual QWERTY keyboard input system. The problem occurs when the users try to use languages other than English. Because the users are required to type their native language or other languages less frequently than English. Voice-based dictating or typing is a solution to this problem. This work focuses on Bangla voice typing. A voice keyboard is a great tool for the Bengali community, especially for people in Bangladesh, India, and other Countries. Usually Bangladeshi uses two font families of two different environments. One font family uses Avro (Unicode) environment another one uses Bijoy (ASCII). At present available Bangla Voice Keyboard is in Unicode. Unfortunately, no such Bangla Voice Keyboard exists to write the widely used SutonnyMJ font of the Bijoy environment which is mainly used for official works. So, there is a need for a Bangla Voice Keyboard that can directly write in SutonnyMJ font when the cursor is placed on any text input field or any kind of word processing software. This work aims to solve the aforementioned problems and offers a solution to write two different font families, and English as well as able to convert the Unicode into Bijoy font so that the users can use this technology interchangeably between Bijoy and Unicode. This proposed work is successfully able to write in Bangla in Unicode and also able to write in Bijoy font with satisfactory accuracy. This Bangla Voice Keyboard is able to produce almost all the complex Bangla letters in SutonnyMJ font which was only possible in Unicode Bangla font. Python programming language is used to develop the entire project. Tkinter Python package is used to design the user interface, keyboard module is used for writing in cursor position. PyAudio is another package that is used for audio data analysis. There are still some minor issues with some characters in SutonnyMJ font. However, this work is able to prove the concept and has open doorways for removing the barrier to typing comfortably in the Bangla language with the help of voice commands.

## TABLE OF CONTENTS

<b>CHAPTER 1 .....</b>	<b>1</b>
INTRODUCTION .....	1
1.1 Overview.....	1
1.2 Research Background.....	3
Comparison of Top Speech to Text Software[7] .....	4
1.3 Contribution of the work.....	5
<b>CHAPTER 2 .....</b>	<b>6</b>
LITERATURE REVIEW AND RELATED WORKS .....	6
2.1 Overview.....	6
<b>CHAPTER 3 .....</b>	<b>10</b>
PROPOSED MODEL.....	10
3.1 Overview.....	10
3.2 Google Cloud Speech API.....	11
3.1.1 Key features.....	11
Speech adaptation .....	11
Domain-specific models .....	11
Easily compare quality.....	11
Speech-to-Text On-Prem .....	11
Speech On-Device .....	11
<b>CHAPTER 4 .....</b>	<b>13</b>
SYSTEM IMPLEMENTATION .....	13
4.1 Overview.....	13
4.2 Character encoding[9].....	13
4.2.1 ASCII.....	13
ASCII advantages and disadvantages .....	15
Advantages .....	15
Disadvantages .....	15
4.2.2 Unicode and UTF-8.....	16
4.3 Character Mapping .....	18
Character Mapping Unicode (UTF-8) To Bijoy (ASCII Encoded).....	18
Complex letter mapping for better view .....	24
<b>CHAPTER 5 .....</b>	<b>33</b>
RESULTS AND DISCUSSION .....	33
5.1 Overview.....	33
<b>CHAPTER 6 .....</b>	<b>37</b>
CONCLUSIONS AND FUTURE WORKS .....	37
6.1 Conclusions .....	37
6.2 Future Works .....	37
<b>REFERENCES .....</b>	<b>38</b>

## List of Figures

<i>Figure 1: Unicode value of Bengali Alphabet (Hexadecimal).....</i>	<i>2</i>
<i>Figure 2: Some paid dictation software.....</i>	<i>4</i>
<i>Figure 3: Speech to Text conversion Model of any particular font.....</i>	<i>10</i>
<i>Figure 4: Working principle of Google Cloud Speech API.....</i>	<i>12</i>
<i>Figure 5: Unicode UTF-8 (Bangla) to Bijoy (ASCII Encoded) .....</i>	<i>23</i>
<i>Figure 6: Complex Bijoy Word Mapping for better view in the document.....</i>	<i>28</i>
<i>Figure 7: List of all possible Complex word in Bangla (Bijoy Font) .....</i>	<i>29</i>
<i>Figure 8: English writing in Microsoft Word Document.....</i>	<i>33</i>
<i>Figure 9: Bangla writing in Microsoft Word Document.....</i>	<i>34</i>
<i>Figure 10: App Searching using Voice Command.....</i>	<i>34</i>
<i>Figure 11: Content searching in YouTube.....</i>	<i>35</i>
<i>Figure 12: Bangla Writing in Microsoft Word in Bijoy Font.....</i>	<i>35</i>
<i>Figure 13: Google Searching using Voice Command.....</i>	<i>36</i>

# Chapter 1

## Introduction

### 1.1 Overview

To type Bangla texts in an editor, Bangla keyboard interfaces, like Bijoy[1] , Proshika [2], Proboton [3], Ekusheyr Shadhinota[4], Avro[5] etc. are used. When a key is pressed, the key code is captured by the interface. The captured codes are mapped to target codes and sent to the editor. For example, if ‘amader’ is typed in an editor, while the Avro phonetic keyboard interface is active, it maps the sequence of keystrokes amader to sequence of (Uni)codes 2438 2478 2494 2470 2503 2480 which represent respectively the Bengali characters, আ ম া দ ে র. Each keyboard interface is related to a keyboard layout which defines which key on the keyboard represents which Bengali character, i.e., the (Uni)code. For most of the layouts, such mapping from keystrokes to target codes is not direct one-to-one. The generated codes by the previously used keyboard interfaces were not standardized, i.e., which code represented which Bangla character was different from interface to interface. This raised a big compatibility problem. For example, the Proshika file was not compatible with Bijoy file unless it was converted. However, all newly released keyboard interfaces follow the Unicode standards, i.e., the keystrokes are mapped to Bangla Unicode texts which are compatible from one interface to another. Because of the standardization, huge Bangla Unicode text will be prevailing in near future. For example, online version of most of the Bangla newspapers have already been started to adopt the Unicode standard. The huge collection of Bangla Unicode text will be used in many ways. Therefore, the analysis and interpretation of the Bangla Unicode text is important. There have a number of Bangla keyboard layouts. We have categorized them into three types. In Standard Layouts we type a word in traditional style as if we are writing it on a paper. Such layouts are the oldest among all types. Using this type of layouts, if we want to write দেয়, we have to press key for ে then for দ and finally for য়. Bijoy, Munir[6], Lekhoni, Gitanjali, Satyajit etc. are such type of layouts. In Unicode Layouts (sometimes called Bangla Phonetic Based Layouts[7])the dependent vowels are typed after the consonant it modifies. For example, to write দেয় we have

# Bengali Alphabet

অ	আ	ই	ঈ	উ	ঊ	ঋ	এ
U+0985	U+0986	U+0987	U+0988	U+0989	U+098A	U+098B	U+098F
ঐ	ও	ঔ	ক	খ	গ	ঘ	ঙ
U+0990	U+0993	U+0994	U+0995	U+0996	U+0997	U+0998	U+0999
চ	ছ	জ	ঝ	ঞ	ট	ঠ	ড
U+099A	U+099B	U+099C	U+099D	U+099E	U+099F	U+09A0	U+09A1
ঢ	ণ	ত	থ	দ	ধ	ন	প
U+09A2	U+09A3	U+09A4	U+09A5	U+09A6	U+09A7	U+09A8	U+09AA
ফ	ব	ভ	ম	য	র	ল	শ
U+09AB	U+09AC	U+09AD	U+09AE	U+09AF	U+09B0	U+09B2	U+09B6
ষ	স	হ	য়	ড়	ঢ়	ৎ	া
U+09B7	U+09B8	U+09B9	U+09DF	U+09DC	U+09DD	U+09CE	U+0981
ং	ঃ	্					
U+0982	U+0983	U+09CD					

Figure 5: Unicode value of Bengali Alphabet (Hexadecimal)[8]

to press key for া, then for ে and finally for ি . UniBijoy (modern style typing)[9], Bangla Unicode[10], National, Munir Unicode, UniJoy, Baishakhi etc are such type of layouts. Transliteration Based Layouts use the English to Bangla transliteration scheme[11]. It uses the English QWERT layout. Using such layouts, if we want to write দেশ, we have to type word like desh or deS, which will then be converted automatically to দেশ . Avro Phonetic, Kickkeys[12], etc., are such type of layout. Since the number of Bangla keyboard layouts is not very few, discussing our work for each of them is not possible within the short scope of this paper.

## 1.2 Research Background

A dictation app is used to transcript documents by speaking. The transcription software has a voice-to-text recognition feature. Application to transcribe instead of typing the document. Dictation software has multiple features such as Advanced Speech Recognition (ASR), Text To Speech (TTS), and Speech Synthesis. Some apps have advanced features such as speaker authentication and Optical Character Recognition (OCR).

A speech recognition app does not just convert voice to text. Some dictation software allows you to dictate and control the Internet browser. Additionally, there is some dictation software that lets to control electronic devices such as the car navigation system. A speech recognition application can halve the time to write a document. On average, users can type up to 30 words per minute. Using dictation software, users can easily transcribe 150 words per minute.

Bijoy Fonts are a None-Unicode Bangla true type font. It is also known as Bangla Legacy Fonts. These Fonts uses English characters to produce Bangla script. You will not be able to see Bangla contents written in this fonts properly if you don't have Bijoy fonts installed on your system.

These are old but most popular Bangla fonts that's why they are called legacy fonts. Generally, these kinds of fonts are used widely to write Bangla in legacy (Bijoy) format.

Bengali writers, Journalists and news editors write their journal, news in Bijoy font. But Bijoy Bengali font is not displayed well. So, they need to convert Unicode for presenting fonts in sophisticated way.

*Comparison of Top Speech to Text Software[13]*

Tool Name	Best For	Platform	Price	Free Trial	Ratings
Dragon Speech Recognition Solutions	Students, legal, health care, and other professionals to transcribe text and share documents with high encryption.	Supports Android, iPhone, PC, and Blackberry devices	Dragon Home for Students \$155 Dragon for Professionals starts at \$116 per year per user	7 days	4/5
Braina	Dictating text using a human language interface on any website or software.	Windows, iOS, and Android devices	Basic Free Braina Pro costs \$49 per year Braina Lifetime \$139	No	5/5
Google Docs Voice Typing	Transcribing text for free on Google Docs online.	PC and Mac devices using Chrome	Free	No	4.5/5
Apple Dictation	Transcribing text for free on Apple devices.	Mac devices	Free	No	4.5/5
Winscribe	Legal, health care, law enforcement, education, and other professionals to dictate text on Android and iPhone devices.	Supports Android, iPhone, PC, and Blackberry devices	Starts at \$284 per user per year	7 days	4/5

*Figure 6: Some paid dictation software*



### 1.3 Contribution of the work

So, the contributions of this work as follows:

- Voice Typing Keyboard in any type of document in English and Bangla (Bijoy (ASCII Encoded), Unicode (UTF-8)) without installing driver package of any particular font layout.

## Chapter 2

### Literature Review and Related Works

#### 2.1 Overview

The use of computers in Bangladesh began in the early 1980s. The attempt to use Bangla on computers began a few years later.

Two components are required to write Bangla on computers. One is a Bangla font and the other is a keyboard driver. These two components are quite different from each other, but the use of one is completely dependent on the other. These two components are considered under the rubric of Bangla software.

Even after two decades have passed since the advent of Bangla computing, there is scant evidence of any worthwhile official bid to substantively enhance the use of Bangla software. Consequently those who use Bangla on computers or those who would like to use Bangla, have been overwhelmed by a bewildering array of countless Bangla software programmes which are the product of myriad unplanned ventures by different developers at different times.

All attempts to develop Bangla software have been made essentially by individuals or private institutions. Many of these are expatriate Bangladeshis. The development of a Bangla font was the unavoidable first step of developing a Bangla software. There is some disagreement over who made the first attempt to develop a Bangla font in Bangladesh. The table below provides a list of Bangla software programmes developed to date. Only the software programmes developed by Bangladeshis or in active interaction with Bangladesh, have been included in the table.

According to news reports, the first research (1982-1986) on Bangla use on computers was conducted under the supervision of Dr. Syed Mahbubur Rahman of the Electrical Engineering Department of BUET. But no information is available on any practical application of this work. Most computers in use at that time were DOS based (MS-DOS) and Macintosh. Due to the practical advantages of Macintosh's graphical interface and the needs of the publishing industry,

in the beginning, the use of Macintosh software was relatively more prevalent. The first Bangla software developed for Mac was called 'Shahidlipi' marketed in 1984. This package of a font and a keyboard driver was developed by Saif ud Doha Shahid. In December 1988, Ananda Computers of Dhaka launched 'Bijoy', a Bangla font and keyboard package for Macintosh. The enterprise was spearheaded by Mostafa Jabbar and Golam Farukh Ahmed. Later, a Windows version for it was developed.

Since Bangla computing was dependent on the publishing industry, no one took the initiative to develop a Bangla software for the DOS platform till 1988. In November 1988, M Shamsul Huq Choudhury of Automation Engineers marketed 'Abaho', a DOS-based package of font and keyboard. Then came 'Onirban'. Later, its Windows version got a fair amount of popularity. Safe Works' Anko and Sohel's 'Barna', made with graphical interface in DOS mode, attracted attention. In 1993, they developed a Windows interface for this and added a Bangla spell-checker for the first time. With the increased popularity of Windows PC and increased use of computers a slew of Bangla software programmes entered the market in the 1990s. In 1993, 'Jatiyo' was developed with government approval. But its use was very limited and it gradually disappeared. During this time, Proshika launched 'Proshika Shabda', an aggregation of a number fonts and keyboard drivers. Later on, a Bangla spell-checker was added to the Proshika Shabda package. In 1994, a number of Bangla software programmes were launched, but none are known to be particularly noteworthy.

Expatriate Bangladeshi computer scientists began to work on using Bangla for computers from the beginning of the 1980s. In 1984, Dr. Abdul Mottaleb of AIT in Thailand developed a Bangla font for DOS. Few details are available about this effort. In 1985, Dr. Muhammad Jafar Iqbal developed a Bangla font for Macintosh in the United States. Besides the standard-sized letters, he developed smaller-sized Bangla letters, similar to the Roman upper and lower case alphabet, for use to form juktakkhor or conjoint consonants. As the 8-bit coding system of the computer does not accommodate all Bangla letters/symbols, he tried to bring about a change in the rules of writing Bangla. In this new system, many traditional juktakkhor or conjoint consonants appeared in new forms. He did not continue further with his efforts to change the traditional writing system of Bangla.

A new chapter began in Bangla software when 'Orcosoft Borno' was launched from the United States in June of 1998. Orcosoft Borno, which was based on the keyboard layout and algorithms designed by Dr. Abdus Shakil, was developed by Tahmid Choudhury. This was a limited-functionality word processor. The basis of the keyboard layout was phonetic, and it was based on a simple, logical set of rules. Orcosoft Borno was later marketed through 'ProtivaSoft' under the name of 'Lekhok' with some added features. After Orcosoft, several Bangla software programmes were developed with a phonetic keyboard layout. These include 'Adarsha Bangla', 'Natural Bangla', 'BornoSoft Bangla2000' and 'Bangla Word'. However, no other software followed a consistent logical system, except perhaps Bangla2000. The Bangla2000, a limited-function Bangla word processor, was launched by BornoSoft and written by US resident student Amirul Islam Manik.

Although there is no complete Bangla software for Linux, several Bangladeshis are working on this on individual initiative. Leading among these is Tanim Ahmed, a Canadian citizen now working in the US. In addition to working on providing system-level support for Linux in Bangla, he is also working on unicode-based open type fonts for Bangla. No known effort in Bangla software separately for Unix platform has been reported. In addition to Bangladeshis, several foreigners are working on developing Bangla software. Robin Upton of Altruists in England is one of them. He has developed 'Ekhushey', an add-on Bangla software, which works exclusively on Microsoft Word. Indian Bangalis from India's West Bengal and abroad have also developed an almost equal number of Bangla software programmes.

The incredible profusion of Bangla software make two things clear: (a) it is not easy to write Bangla on computers, and (b) many developers have tried to make the use of Bangla easy in this medium. Every developer claims their own product to be easier than the others. But it is fair to say that no one has succeeded in proving this. The main problem with the hitherto launched Bangla programmes is that these programmes require a user to practice and master a new keyboard. It is inevitable that other than professional Bangla typists, no one will expend time to master a new keyboard for Bangla in addition to the Roman keyboard. Consequently, although there are over two-dozen Bangla software programmes available in the market, less than five percent of computer users in Bangladesh can write Bangla on the computer. It is so expensive and time-consuming to master the Bangla keyboard that at one point, some software were

developed to allow typing Bangla on the computer by clicking a mouse on the Bangla alphabet on the monitor. These efforts underscore the widespread inability to write Bangla by using a keyboard. This also suggests that typing Bangla on a computer is exclusively the job of a professional Bangla typist.

The large number of Bangla characters is often blamed for being the main obstacle to the development of an easy Bangla software. But countries like China and Japan have taken steps to type Chinese and Japanese on the computer comfortably, using the knowledge of the 26 keys on the Roman keyboard, notwithstanding the fact that each language has over 1,000 characters. Almost all languages have developed a standardized transliteration scheme for representing words in their language in Roman script. Such a scheme, which can be mastered in an hour, enables even a foreigner without the ability to read the language to type it on the computer using only the pronunciation. However, this had not been the case in Bangladesh with Bangla software.

US resident Dr. Abdus Shakil, a physician-scientist by profession and the founder of BornoSoft, has devised such a logical and comprehensive system after years of research in Bangla. The Bangla2000 software launched by BornoSoft is based on that system. Anyone, even a foreigner, can master the system in an hour and can fluently write Bangla on computers using 26 lower-case keys of the Roman keyboard.

Since the beginning of the use of computers in Bangladesh, one Bangla software after another has been launched. Although none has been able to satisfy the users, Bijoy ranks on the top of the list of number of users. Most professional Bangla typists use Bijoy. Next in popularity is probably the Munir keyboard (based on the Munir Optima typewriter) that is included in a number of packages. On the other hand, BornoSoft Bangla2000 is the Bangla software of choice for Bangla users in Europe and North America. Among the non-professional Bangla typists in Bangladesh, BornoSoft's Bangla2000 is gradually gaining popularity[14].

# Chapter 3

## Proposed Model

### 3.1 Overview

The aim of the work is developing a speech to text transcriber model using Google Cloud Speech API that needs the internet connection or using locally trained language model that does not need the internet connection. But using Google Cloud Speech API the accuracy is almost 100% where the accuracy for locally trained language model is very less than that. So this paper covers only using the Google Cloud Speech API.

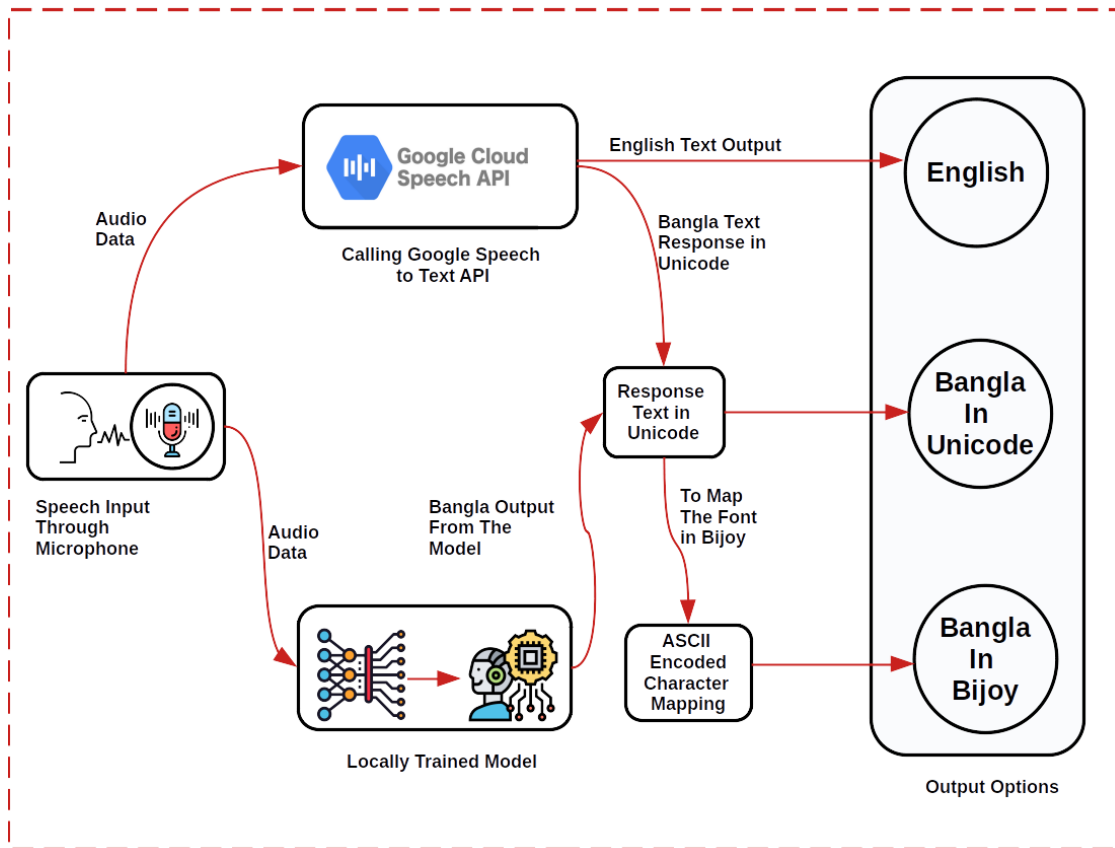


Figure 7: Speech to Text conversion Model of any particular font

## 3.2 Google Cloud Speech API

### 3.1.1 Key features

#### *Speech adaptation*

Provide hints to boost the transcription accuracy of rare and domain-specific words or phrases. By using classes to automatically convert spoken numbers into addresses, years, currencies, and more.

#### *Domain-specific models*

By selecting for voice control, phone call, and video transcription optimized for domain-specific quality requirements.

#### *Easily compare quality*

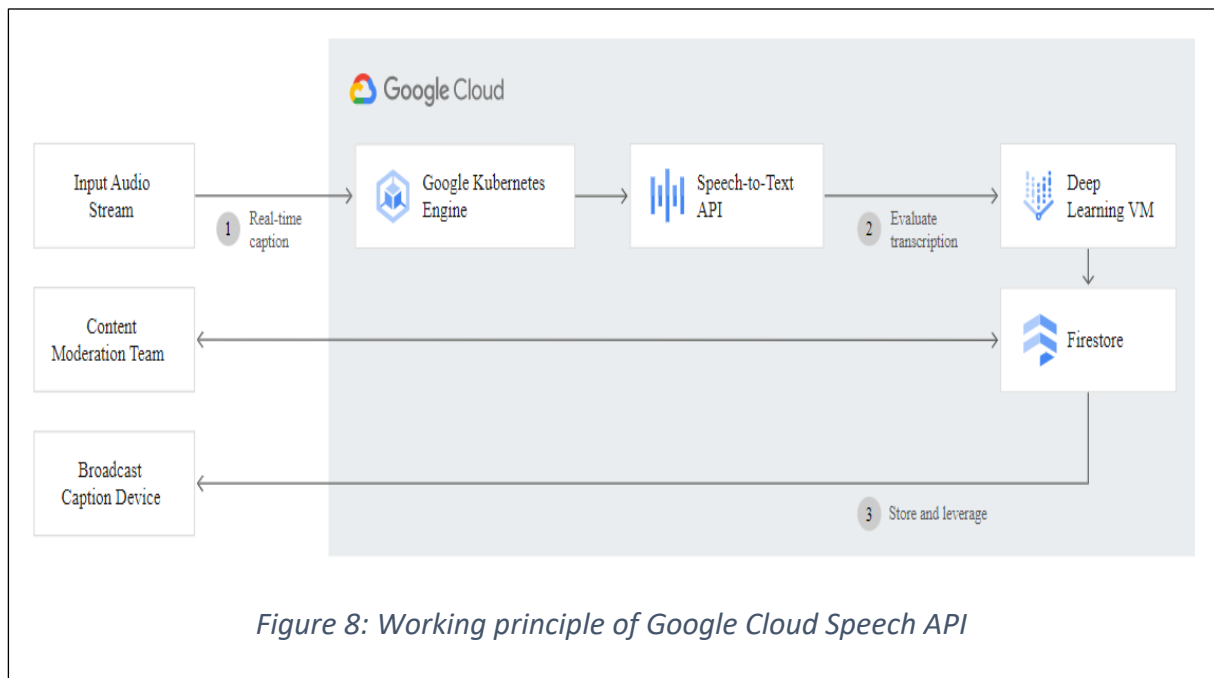
Experiment on the speech audio with easy-to-use user interface. Different configurations to optimize quality and accuracy.

#### *Speech-to-Text On-Prem*

Maintain control over infrastructure and protected speech data by leveraging Google's speech recognition technology on-premises, in the private data centers.

#### *Speech On-Device*

Running Google Cloud's speech algorithms locally on any device, regardless of internet connectivity. Promise users that their voice data will never leave their device.





# Chapter 4

## System Implementation

### 4.1 Overview

This chapter covers the system development process in details. Character mapping is a great technique for converting font from one layout to another layout. Since this system does not require the keyboard interface of a particular font, so we need to map the characters differently. ASCII encoding of Unicode Bengali font for getting the Bijoy font is complex but this system uses it.

### 4.2 Character encoding[15]

ASCII and UTF-8 are two modern text encoding systems.

#### 4.2.1 ASCII

The **American Standard Code for Information Interchange** (ASCII) was developed to create an international standard for encoding the Latin alphabet. In 1963, ASCII was adopted so information could be interpreted between computers; representing lower and upper letters, numbers, symbols, and some commands. Because ASCII is encoded using ones and zeros, the base 2 number system, it uses seven bits. Seven bits allow  $2$  to the power of  $7 = 128$  possible combinations of digits to encode a character.

ASCII therefore made sure that 128 important characters could be encoded:

How encoding ASCII works

- You already know how to convert between denary and binary numbers
- You now need to turn letters into binary numbers
- Every character has a corresponding denary number (for example,  $A \rightarrow 65$ )

- ASCII uses 7 bits or 8 bits
- We use the first 7 columns of the conversion table to create 128 different numbers (from 0 to 127)

For example, 1000001 gives us the number 65 ( $64 + 1$ ), which corresponds to the letter 'A'.

64	32	16	8	4	2	1
1	0	0	0	0	0	1

Here's how 'HELLO' is encoded in ASCII in binary:

Latin Character	ASCII
H	1001000
E	1000101
L	1001100
L	1001100
O	1001111

## ASCII advantages and disadvantages

After more than half a century of use, the advantages and disadvantages of using ASCII character encoding are well understood. That is one of the encoding format's great strengths.

### *Advantages*

- **Universally accepted.** ASCII character encoding is universally understood. Except for the IBM mainframes that use EBCDIC encoding, it is universally implemented in computing through the Unicode standard. Unicode character encoding replaces ASCII encoding, but it is backward-compatible with ASCII.
- **Compact character encoding.** Standard codes can be expressed in 7 bits. This means data that can be expressed in the standard ASCII character set requires only as many bytes to store or send as the number of characters in the data.
- **Efficient for programming.** The character codes for letters and numbers are well adapted to programming techniques for manipulating text and using numbers for calculations or storage as raw data.

### *Disadvantages*

- **Limited character set.** Even with extended ASCII, only 255 distinct characters can be represented. The characters in a standard character set are enough for English language communications. But even with the diacritical marks and Greek letters supported in extended ASCII, it is difficult to accommodate languages that do not use the Latin alphabet.
- **Inefficient character encoding.** Standard ASCII encoding is efficient for English language and numerical data. Representing characters from other alphabets requires more overhead such as escape codes.

### 4.2.2 Unicode and UTF-8

Because ASCII encodes characters in 7 bits, moving to 8-bit computing technology meant there was one extra bit to be used. With this extra digit, **Extended ASCII** encoded up to 256 characters. However, the problem that developed was that countries that used different languages did different things with this extra capacity for encoding. Many countries added their own additional characters, and different numbers represented different characters in different languages. Japan even created multiple systems of encoding Japanese depending on the hardware, and all of these methods were incompatible with each other. So when a message was sent from one computer to another, the received message could become garbled and unreadable; the Japanese character encoding systems were so complex that even when a message was sent from one type of Japanese computer to another, something called ‘Mojibake’ would happen:

The problem of incompatible encoding systems became more urgent with the invention of the World Wide Web, as people shared digital documents all over the world, using multiple languages. To address the issue, the Unicode Consortium established a universal encoding system called Unicode. Unicode encodes more than 100000 characters, covering all the characters you would find in most languages. Unicode assigns each characters a specific number, not to a binary digit. But there were some issues with this, for example:

1. To encode 100000 characters, around 32 binary digits would be required. Unicode uses ASCII for the English language, so A is still 65. However, encoded in 32 bits, the binary representation for the letter A would be 000000000000000000000000000000001000001. This wastes a lot of valuable space!
2. Many older computers interpret eight zeros in a row (a null) as the end of a string of characters. So these computers wouldn't send any characters that came after eight zeros in a row (they wouldn't send an A if it was represented as 000000000000000000000000000000001000001).

The Unicode encoding method **UTF-8** solves these problems:

Up to character number 128, the regular ASCII value is used (so for example A is 01000001)– For any character beyond 128, UTF-8 separates the code into two bytes and adding ‘110’ to

the start of first byte to show that it is a beginning byte, and '10' to the start of second byte to show that it follows the first byte.

So, for each character beyond number 128, you have two bytes:

```
[110xxxxx] [10xxxxxx]
```

And you just fill in the binary for the number in between:

```
[11000101] [10000101] (that's the number 325 → 00101000101)
```

This works for the first 2048 characters. For characters beyond that, one more '1' is added at the beginning of the first byte and a third byte is also used:

```
[1110xxxx] [10xxxxxx] [10xxxxxx]
```


This gives you 16 spaces for binary code. In this manner, UTF-8 goes up to four bytes:

```
[11110xxx] [10xxxxxx] [10xxxxxx] [10xxxxxx]
```

In this way, UTF-8 avoids the problems mentioned above as well as needing an index, and it lets you decode characters from the binary form backwards (i.e. it is backwards-compatible).

### 4.3 Character Mapping

*Character Mapping Unicode (UTF-8) To Bijoy (ASCII Encoded)*



1	" " : " " ,
2	" " : " " ,
3	"' " : "ô" ,
4	"' " : "õ" ,
5	"' " : "ò" ,
6	"' " : "ó" ,
7	"্য" : "a " ,
8	"স্প্র" : "xcö" ,
9	"র্য" : "i " ,
10	"ক্ষ্ম" : "²" ,
11	"ক্ক" : "o" ,
12	"ক্ট" : "±" ,
13	"ক্ত" : "³" ,
14	"ক্ক" : "Ki" ,
15	"ক্ক" : "⁻E" ,
16	"ক্র" : "μ" ,
17	"ক্ক" : "K-" ,
18	"ক্ষ" : "¶" ,
19	"ক্স" : "•" ,
20	"গু" : " ,

Figure 9a

21	"ধ্ব" : "»" ,
22	"গ্ন" : "Mœ" ,
23	"গ্ম" : "M¥" ,
24	"গ্ন" : "M•" ,
25	"গ্র" : "M=y" ,
26	"ক্ক" : "¼" ,
27	"ঔক্ষ" : "•¶" ,
28	"জ্ব" : "•L" ,
29	"জ্জ" : "½" ,
30	"ঔঘ" : "•N" ,
31	"চ্ছ" : "Qi" ,
32	"চ্চ" : "P" ,
33	"চ্ছ" : "Q" ,
34	"চ্ছ" : "T" ,
35	"জ্জ্ব" : "¾i" ,
36	"জ্জ" : "¾" ,
37	"জ্ব" : "À" ,
38	"জ্জ" : "Á" ,
39	"জ্ব" : "Ri" ,
40	"ধ্ব" : "Â" ,

Figure 5b

41 "ঞ": "Ã",  
 42 "ঞ": "Ä",  
 43 "ঞ": "Å",  
 44 "ঞ": "Æ",  
 45 "ঞ": "Uï",  
 46 "ঞ": "U¥",  
 47 "ঞ": "Ç",  
 48 "ঞ": "È",  
 49 "ঞ": "É",  
 50 "ঞ": "Ý",  
 51 "ঞ": "Ê",  
 52 "ঞ": "Š' ",  
 53 "ঞ": "Y^",  
 54 "ঞ": "Ěi",  
 55 "ঞ": "Ě",  
 56 "ঞ": "İ",  
 57 "ঞ": "Zæ",  
 58 "ঞ": "Z¥",  
 59 "ঞ": "Š-i",  
 60 "ঞ": "Zi",

Figure 5c

61 "থ": "\_i",  
 62 "দগ": "~M",  
 63 "দঘ": "~N",  
 64 "দ": "İ",  
 65 "দ্ব": "x",  
 66 "দ্ব": "~i",  
 67 "দ্ব": "Ø",  
 68 "দ্ব": "™¢",  
 69 "দ্ব": "Ù",  
 70 "দ্ব": "`\_a'",  
 71 "দ্ব": "aÿ",  
 72 "দ্বম": "a¥",  
 73 "দ্ব": ">U",  
 74 "দ্ব": "Ú",  
 75 "দ্ব": "Û",  
 76 "দ্ব": "šì",  
 77 "দ্ব": "š-",  
 78 "দ্ব": "-i",  
 79 "দ্ব": "Î",  
 80 "দ্ব": "š' ",

Figure 5d



81 "ন্দ": ">`",  
82 "ন্ব": ">Ø",  
83 "ন্ধ": "Ü",  
84 "ন্ন": "bœ",  
85 "ষ": "š^",  
86 "ন্ম": "b¥",  
87 "প্ট": "p",  
88 "প্ত": "ß",  
89 "প্প": "cœ",  
90 "প্প": "à",  
91 "প্প": "c•",  
92 "প্প": "á",  
93 "ফ্ফ": "d-",  
94 "জ্জ": "â",  
95 "দ্দ": "ã",  
96 "ক্ক": "ä",  
97 "ব্ব": "eÿ",  
98 "ব্ব": "e•",  
99 "ভ্ভ": "å",  
100 "ম্ম": "gœ",

Figure 5e

101 "ক্ষ": "xú",  
102 "ক্ষ": "ç",  
103 "ষ": "x^",  
104 "ভ্ভ": "xç",  
105 "ম্ম": "xf",  
106 "ম্ম": "xš",  
107 "ম্ম": "x•",  
108 "ঞ": "a",  
109 "রু": "i'",  
110 "রু": "if",  
111 "ক্ক": "é",  
112 "ব্ব": "ê",  
113 "ল্ট": "ë",  
114 "ল্ট": "ì",  
115 "ব্ব": "í",  
116 "ক্ষ": "î",  
117 "ব্ব": "j|",  
118 "ম্ম": "j¥",  
119 "ম্ম": "j•",  
120 "শু": "i",

Figure 5f



121	"𑌕": "ø",
122	"𑌖": "kæ",
123	"𑌗": "kǐ",
124	"𑌘": "kɤ",
125	"𑌙": "k•",
126	"𑌚": "®<",
127	"𑌛": "®E",
128	"𑌜": "ó",
129	"𑌝": "ô",
130	"𑌞": "ò",
131	"𑌟": "®ú",
132	"𑌠": "õ",
133	"𑌡": "®§",
134	"𑌢": "⁻<",
135	"𑌣": "÷",
136	"𑌤": "ö",
137	"𑌥": "⁻—",
138	"𑌦": "⁻' ",
139	"𑌧": "⁻' ",
140	"𑌨": "mæ",

Figure 5g

141	"𑌩": "⁻ú",
142	"𑌪": "ù",
143	"𑌫": "⁻^",
144	"𑌬": "⁻§",
145	"𑌭": "⁻•",
146	"𑌮": "û",
147	"𑌯": "nè",
148	"𑌰": "nÿ",
149	"𑌱": "ý",
150	"𑌲": "p",
151	"𑌳": "n¬",
152	"𑌴": "ü",
153	"𑌵": "©",
154	"𑌶": "«",
155	"𑌷": "⁻",
156	"𑌸": "&",
157	"𑌹": "A",
158	"𑌺": "Av",
159	"𑌻": "B",
160	"𑌼": "C",

Figure 5h

161 "উ": "D",  
 162 "ঊ": "E",  
 163 "ঋ": "F",  
 164 "এ": "G",  
 165 "ঐ": "H",  
 166 "ও": "I",  
 167 "ঔ": "J",  
 168 "ক": "K",  
 169 "খ": "L",  
 170 "গ": "M",  
 171 "ঘ": "N",  
 172 "ঙ": "O",  
 173 "চ": "P",  
 174 "ছ": "Q",  
 175 "জ": "R",  
 176 "ঝ": "S",  
 177 "ঞ": "T",  
 178 "ট": "U",  
 179 "ঠ": "V",  
 180 "ড": "W",

Figure 5i

181 "ঢ": "X",  
 182 "ণ": "Y",  
 183 "ত": "Z",  
 184 "থ": "\_",  
 185 "দ": "`",  
 186 "ধ": "a",  
 187 "ন": "b",  
 188 "প": "c",  
 189 "ফ": "d",  
 190 "ব": "e",  
 191 "ভ": "f",  
 192 "ম": "g",  
 193 "য": "h",  
 194 "র": "i",  
 195 "ল": "j",  
 196 "শ": "k",  
 197 "ষ": "l",  
 198 "স": "m",  
 199 "হ": "n",  
 200 "ড়": "o",

Figure 5j

201	"ঢ়": "p",
202	"ষ": "q",
203	"৙": "r",
204	"ং": "s",
205	"ঃ": "t",
206	"ঁ": "u",
207	"০": "0",
208	"১": "1",
209	"২": "2",
210	"৩": "3",
211	"৪": "4",
212	"৫": "5",
213	"৬": "6",
214	"৭": "7",
215	"৮": "8",
216	"৯": "9",
217	"+": "+",
218	"-": "-",
219	"x": "x",
220	"*": "*",

Figure 5k


221	"/": "/",
222	"%": "%",
223	"=": "=",
224	",": ",",
225	".": ".",
226	"া": "v",
227	"ি": "w",
228	"ী": "x",
229	"ু": "y",
230	"ূ": "~",
231	"্": "...",
232	"ে": "†",
233	"ৈ": "‰",
234	"ৌ": "Š",
235	"ো": "†v",
236	"ৌ": "†Š",
237	"\n": "\n"

Figure 5l

Figure [5a-5l]: Unicode UTF-8  
(Bangla) to Bijoy (ASCII  
Encoded) Mapping

Bijoy font have some issue with complex word (More than one letter in one syllable). There needs another mapping.

*Complex letter mapping for better view*



1	"&i&h": "ᳵ",
2	"g&c&i": "᳚᳚᳚",
3	"i&h": "ᳵ",
4	"K&l&g": "²",
5	"K&K": "°",
6	"K&U": "±",
7	"K&Z": "³",
8	"K&e": "᳚᳚",
9	"m&K&i": "᳚᳚᳚",
10	"K&i": "᳚",
11	"K&j": "᳚᳚",
12	"K&l": "᳚",
13	"K&m": "᳚",
14	"My": "᳚",
15	"M&a": "᳚᳚",

*Figure 6a*

16	"M&b": "᳚᳚",
17	"M&g": "᳚᳚",
18	"M&j": "᳚᳚",
19	"M&iy": "᳚᳚᳚",
20	"O&K": "᳚",
21	"O&K&l": "᳚᳚",
22	"O&L": "᳚᳚",
23	"O&M": "᳚",
24	"O&N": "᳚᳚",
25	"P&Q&e": "᳚᳚᳚",
26	"P&P": "᳚᳚",
27	"P&Q": "᳚᳚",
28	"P&T": "᳚᳚",
29	"R&R&e": "᳚᳚᳚",
30	"R&R": "᳚᳚",

*Figure 6b*



31 "R&S": "À",  
 32 "R&T": "Á",  
 33 "R&e": "Rì",  
 34 "T&P": "Â",  
 35 "T&Q": "Ã",  
 36 "T&R": "Ä",  
 37 "T&S": "Å",  
 38 "U&U": "Æ",  
 39 "U&e": "Uì",  
 40 "U&g": "Uƒ",  
 41 "W&W": "Ç",  
 42 "Y&U": "È",  
 43 "Y&V": "É",  
 44 "b&m": "Ý",  
 45 "Y&W": "Ê",

Figure 6c

46 "b&Zy": "š'",  
 47 "Y&e": "Y^",  
 48 "Z&Z&e": "Ëì",  
 49 "Z&Z": "Ë",  
 50 "Z&\_": "Ì",  
 51 "Z&b": "Zœ",  
 52 "Z&g": "Zƒ",  
 53 "b&Z&e": "š-ì",  
 54 "Z&e": "Zì",  
 55 "\_&e": "\_ì",  
 56 "`&M": "˜M",  
 57 "`&N": "˜N",  
 58 "`&`": "Ï",  
 59 "`&a": "x",  
 60 "`&e": "Ø",

Figure 6d

61 "`&f": "™¢",  
 62 "`&g": "Ù",  
 63 "`&iy": "à",  
 64 "a&e": "aÿ",  
 65 "a&g": "aƒ",  
 66 "b&U": "›U",  
 67 "b&V": "Ú",  
 68 "b&W": "Û",  
 69 "b&Z&i": "šž",  
 70 "b&Z": "š–",  
 71 "m&Z&i": "˘ž",  
 72 "Z&i": "Î",  
 73 "b&\_": "š'",  
 74 "b&`": "›`",  
 75 "b&`&e": "›Ø",

Figure 6e

76 "b&a": "Ü",  
 77 "b&b": "bœ",  
 78 "b&e": "š^",  
 79 "b&g": "bƒ",  
 80 "c&U": "P",  
 81 "c&Z": "ß",  
 82 "c&b": "cœ",  
 83 "c&c": "à",  
 84 "c&j": "c•",  
 85 "c&m": "á",  
 86 "d&j": "d–",  
 87 "e&R": "â",  
 88 "e&`": "ă",  
 89 "e&a": "ä",  
 90 "e&e": "eÿ",

Figure 6f

91 "e&j": "e•",  
 92 "f&i": "å",  
 93 "g&b": "gæ",  
 94 "g&c": "ɰú",  
 95 "g&d": "ç",  
 96 "g&e": "ɰ^",  
 97 "g&f": "ɰç",  
 98 "g&f&i": "ɰf",  
 99 "g&g": "ɰ§",  
 100 "g&j": "ɰ•",  
 101 "&i": "«",  
 102 "iy": "i'",  
 103 "i~": "if",  
 104 "j&K": "é",  
 105 "j&M": "ê",

Figure 6g

106 "j&U": "ë",  
 107 "j&W": "ì",  
 108 "j&c": "í",  
 109 "j&d": "î",  
 110 "j&e": "j!",  
 111 "j&g": "j‡",  
 112 "j&j": "j•",  
 113 "ky": "ï",  
 114 "k&P": "ð",  
 115 "k&b": "kæ",  
 116 "k&e": "k!",  
 117 "k&g": "k‡",  
 118 "k&j": "k•",  
 119 "l&K": "℔<",  
 120 "l&K&i": "℔£",

Figure 6h

121 "l&U": "ó",  
 122 "l&V": "ô",  
 123 "l&Y": "ò",  
 124 "l&c": "®ú",  
 125 "l&d": "õ",  
 126 "l&g": "®§",  
 127 "m&K": "˘˂",  
 128 "m&U": "÷",  
 129 "m&L": "ö",  
 130 "m&Z": "˘—",  
 131 "m&Zy": "˘' ",  
 132 "m&\_": "˘' ",  
 133 "m&b": "mæ",  
 134 "m&c": "˘ú",  
 135 "m&d": "ù",

Figure 6i

136 "m&e": "˘^",  
 137 "m&g": "˘§",  
 138 "m&j": "˘•",  
 139 "ny": "û",  
 140 "n&Y": "nè",  
 141 "n&e": "nÿ",  
 142 "n&b": "ý",  
 143 "n&g": "þ",  
 144 "n&j": "n¬",  
 145 "n...": "ü",  
 146 "i&": "©",  
 147 "&h": "˘",

Figure 6j

Figure [6a-6j]: Complex Bijoy Word Mapping for better view in the document





```

1 '&i&h', 'g&c&i', 'i&h', 'K&l&g', 'K&K', 'K&U', 'K&Z', 'K&e', 'm&K&i', 'K&i',
2 'K&j', 'K&l', 'K&m', 'My', 'M&a', 'M&b', 'M&g', 'M&j', 'M&iy', 'O&K', 'O&K&l',
3 'O&L', 'O&M', 'O&N', 'P&Q&e', 'P&P', 'P&Q', 'P&T', 'R&R&e', 'R&R', 'R&S', 'R&T',
4 'R&e', 'T&P', 'T&Q', 'T&R', 'T&S', 'U&U', 'U&e', 'U&g', 'W&W', 'Y&U', 'Y&V', 'b&m',
5 'Y&W', 'b&Zy', 'Y&e', 'Z&Z&e', 'Z&Z', 'Z&_', 'Z&b', 'Z&g', 'b&Z&e', 'Z&e', '._&e',
6 '&M', '&N', '&', '&a', '&e', '&f', '&g', '&iy', 'a&e', 'a&g', 'b&U', 'b&V',
7 'b&W', 'b&Z&i', 'b&Z', 'm&Z&i', 'Z&i', 'b&_', 'b&', 'b&&e', 'b&a', 'b&b', 'b&e',
8 'b&g', 'c&U', 'c&Z', 'c&b', 'c&c', 'c&j', 'c&m', 'd&j', 'e&R', 'e&', 'e&a', 'e&e',
9 'e&j', 'f&i', 'g&b', 'g&c', 'g&d', 'g&e', 'g&f', 'g&f&i', 'g&g', 'g&j', '&i', 'iy',
10 'i~', 'j&K', 'j&M', 'j&U', 'j&W', 'j&c', 'j&d', 'j&e', 'j&g', 'j&j', 'ky', 'k&P',
11 'k&b', 'k&e', 'k&g', 'k&j', 'l&K', 'l&K&i', 'l&U', 'l&V', 'l&Y', 'l&c', 'l&d',
12 'l&g&K', 'm&U', 'm&L', 'm&Z', 'm&Zy', 'm&_', 'm&b', 'm&c', 'm&d', 'm&e', 'm&g', 'm&j',
13 'ny', 'n&Y', 'n&e', 'n&b', 'n&g', 'n&j', 'n...', 'i&', '&h'

```

Figure 7: List of all possible Complex word in Bangla (Bijoy Font)

## Algorithm

Function **getRawBijoyFromUnicode** (Unicode):

```
bijoy = "";  
for i from 0 to length (Unicode) do  
    bijoy += Unicode_to_Bijoy(Unicode[i]) // from table-1  
return bijoy
```

// Optimization of juktoBorno Occurance

Function **getIndicesOf**(str, caseSensitive = true):

```
startIndex, index, indices = [], karPos = []  
if not caseSensitive:  
    str = str.toLowerCase()  
    searchStr = searchStr.toLowerCase()  
  
for i from 0 to length(juktoBorno):  
    searchStrLen = length(juktoBorno[i])  
    startIndex = 0  
    if searchStrLen == 0:  
        return []  
    while ((index = str.indexOf(juktoBorno[i], startIndex)) > -1):  
        indices.push(index)  
        startIndex = index + searchStrLen  
        ch = str[startIndex]  
        if (isSingle(ch)):  
            karPos.push(startIndex)  
return karPos
```

function **isSingle**(ch):

```
return ch == 'w' or ch == '†' or ch == '%'
```

```
// Fixing Kar With Not JuktoBorno
```

```
Function fixingKar(rawBijoyText, karPos):
```

```
    bijoy = ""
```

```
    for i from 0 to length(rawBijoyText):
```

```
        if(isSingle(rawBijoyText[i+1]) and !karPos.includes(i+1))
```

```
            bijoy += rawBijoyText[i+1] + rawBijoyText[i]
```

```
            i++
```

```
        else:
```

```
            bijoy += rawBijoyText[i]
```

```
    return bijoy
```

```
function fixingJuktoBornoWithKar(str):
```

```
    startIndex, index, indices = [], karPos = []
```

```
    for(let i=0; i<juktoBorno_len; i++) {
```

```
        var searchStrLen = juktoBorno[i].length;
```

```
        startIndex = 0
```

```
        if (searchStrLen == 0):
```

```
            return [];
```

```
        while ((index = str.indexOf(juktoBorno[i], startIndex)) > -1):
```

```
            // Found Match
```

```
            indices.push(index)
```

```
            startIndex = index + searchStrLen
```

```
            ch = str[startIndex]
```

```

if(isSingle(ch)):
    if(juktoBorno[i] == 'Y&W'):
        str = str.substring(0, index) + ch + juktoBorno[i] + str.substring(startIndex+1)

    else:
        str = str.substring(0, index) + ch + juktoBorno_mapping[juktoBorno[i]] +
        str.substring(startIndex+1)
    else:
        if(juktoBorno[i] == 'Y&W'):
            str = str.substring(0, index) + ch + juktoBorno[i] + str.substring(startIndex+1)
        else:
            str = str.substring(0, index) + juktoBorno_mapping[juktoBorno[i]] +
            str.substring(startIndex)

        startIndex = startIndex - (juktoBorno_mapping[juktoBorno[i]].length -
        juktoBorno[i].length)

return str

```

**/\* Function Call \*/**

RawBijoy = **getRawBijoyFromUnicode** (RandomUnicodeText)

Record = **getIndicesOf** (RawBijoy)

RawBijoy = **fixingKar**(Record)

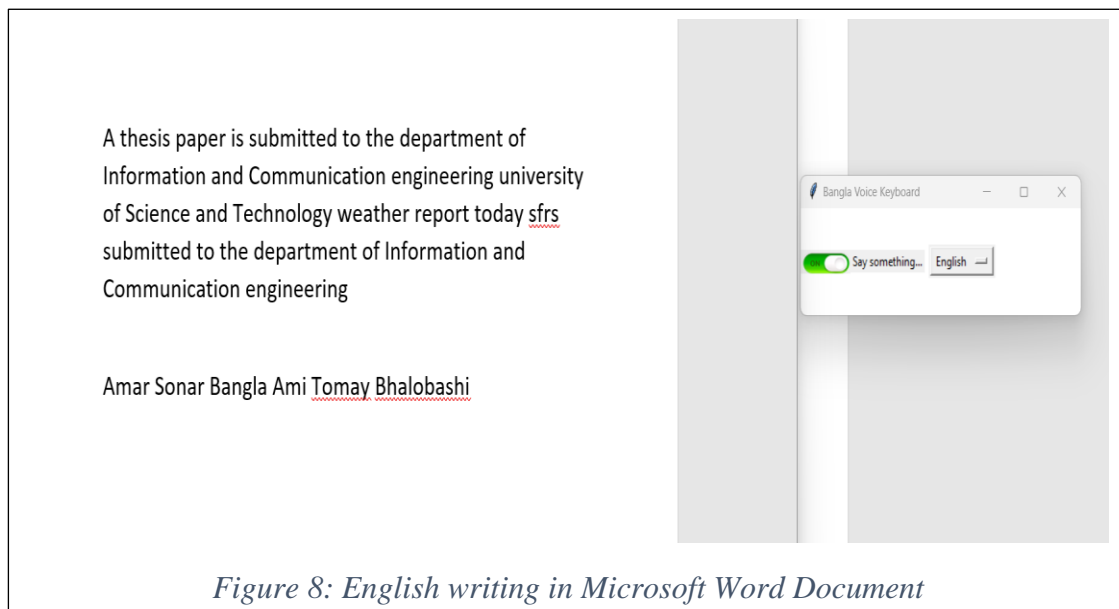
bijoyText = **fixingJuktoBornoWithKar**(RawBijoy)

# Chapter 5

## Results and Discussion

### 5.1 Overview

This thesis based on the development of Bangla voice keyboard. User speech will be parsed into text using the Google Cloud Speech API. Then if the user select the language option to English then the API response directly can be written where the cursor current position is. If the user select the language option Bangla UTF-8 then the response will be directly Unicode Bangla. But if the user select the language option to Bijoy then the response text need to Encode using the ASCII.



A thesis paper is submitted to the department of  
Information and Communication engineering university  
of Science and Technology weather report today sfrs  
submitted to the department of Information and  
Communication engineering

Amar Sonar Bangla Ami Tomay Bhalobashi

আমার সোনার বাংলা আমি তোমায় ভালোবাসি

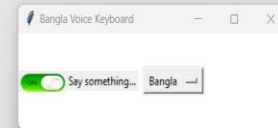


Figure 9: Bangla writing in Microsoft Word Document

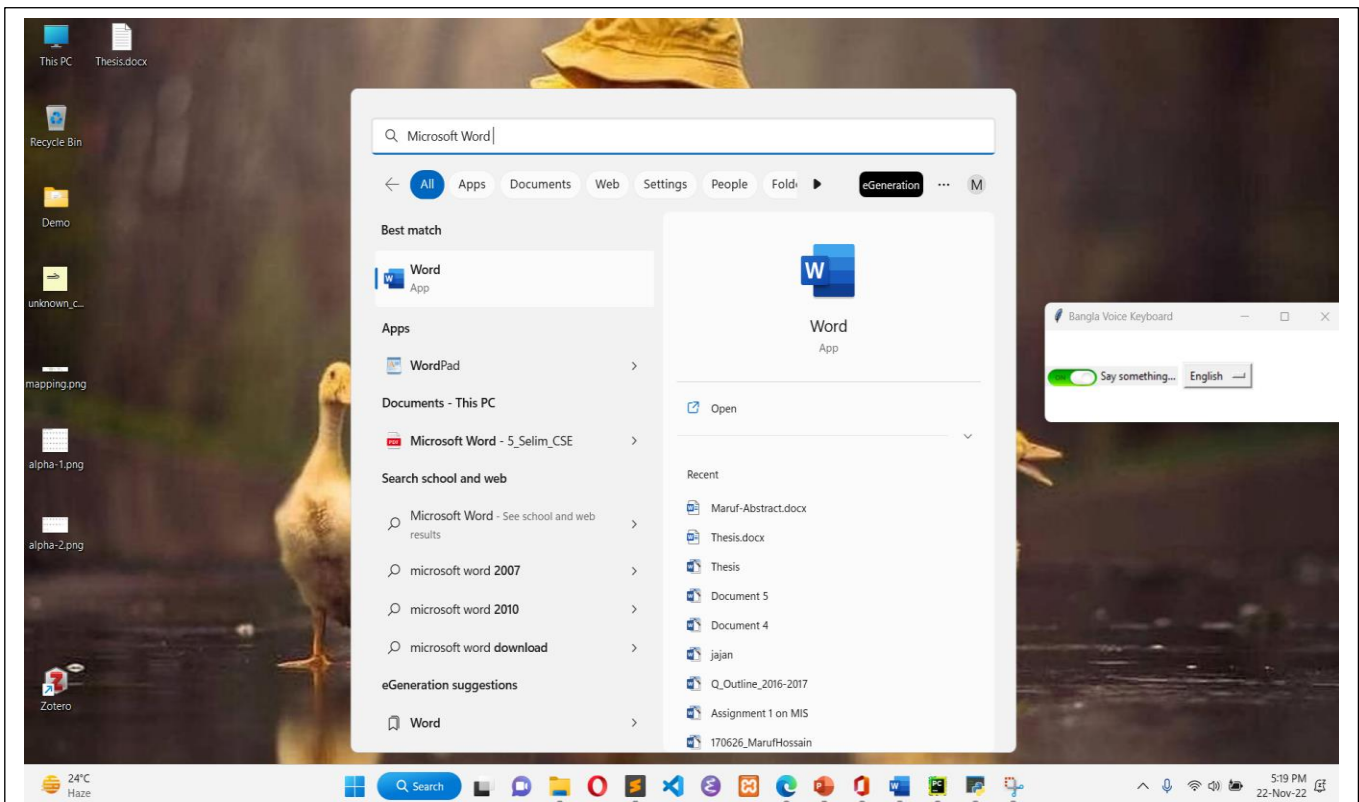


Figure 10: App Searching using Voice Command

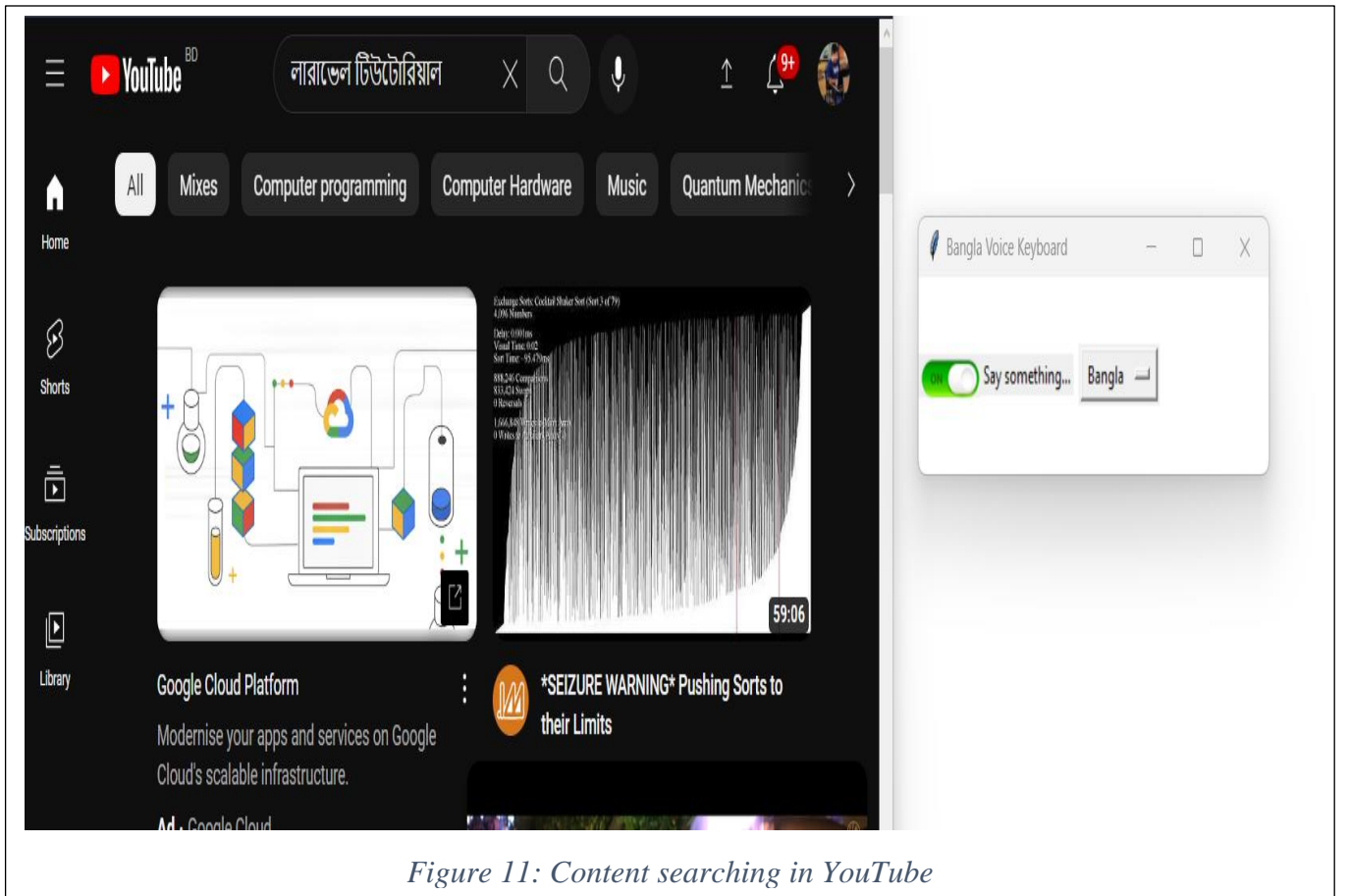
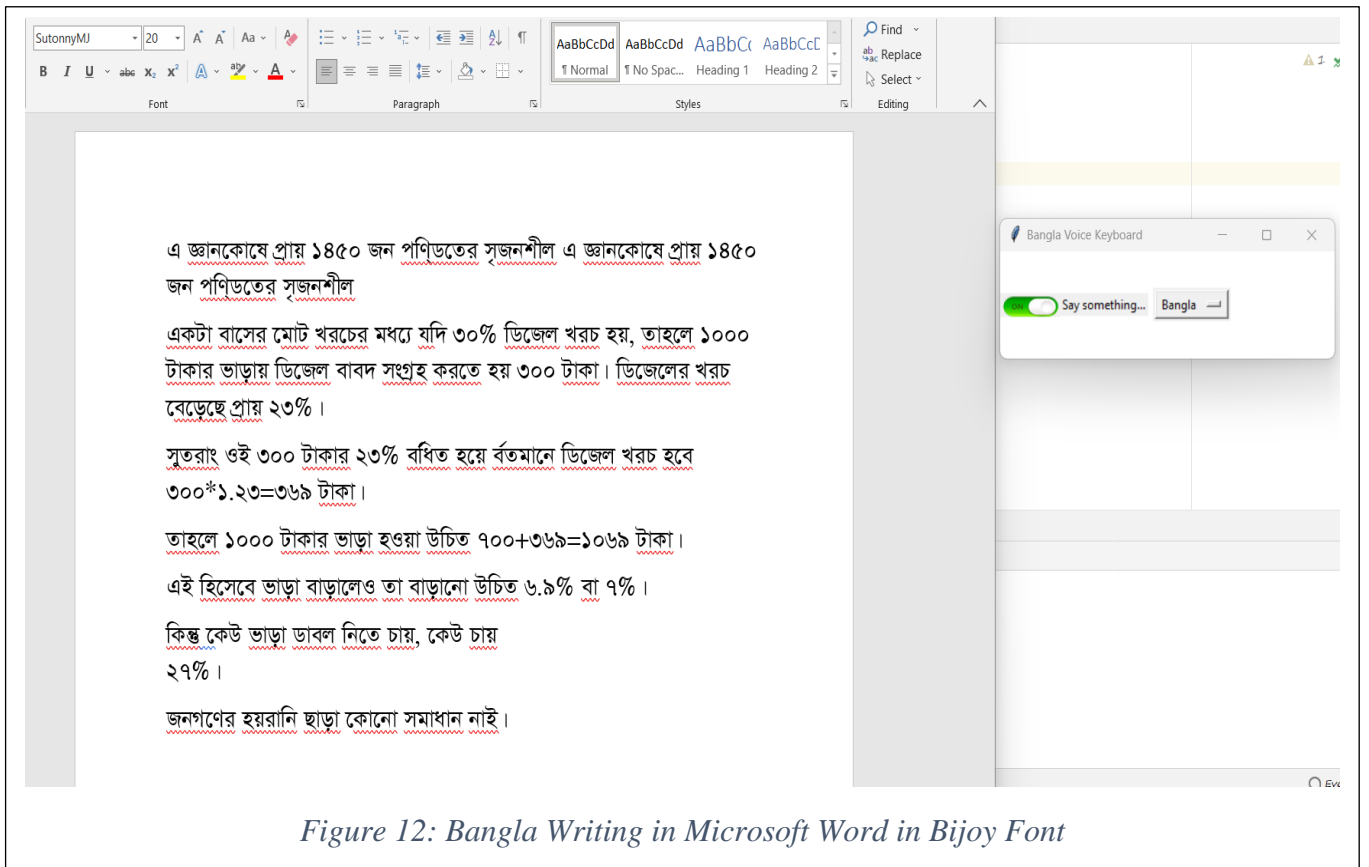
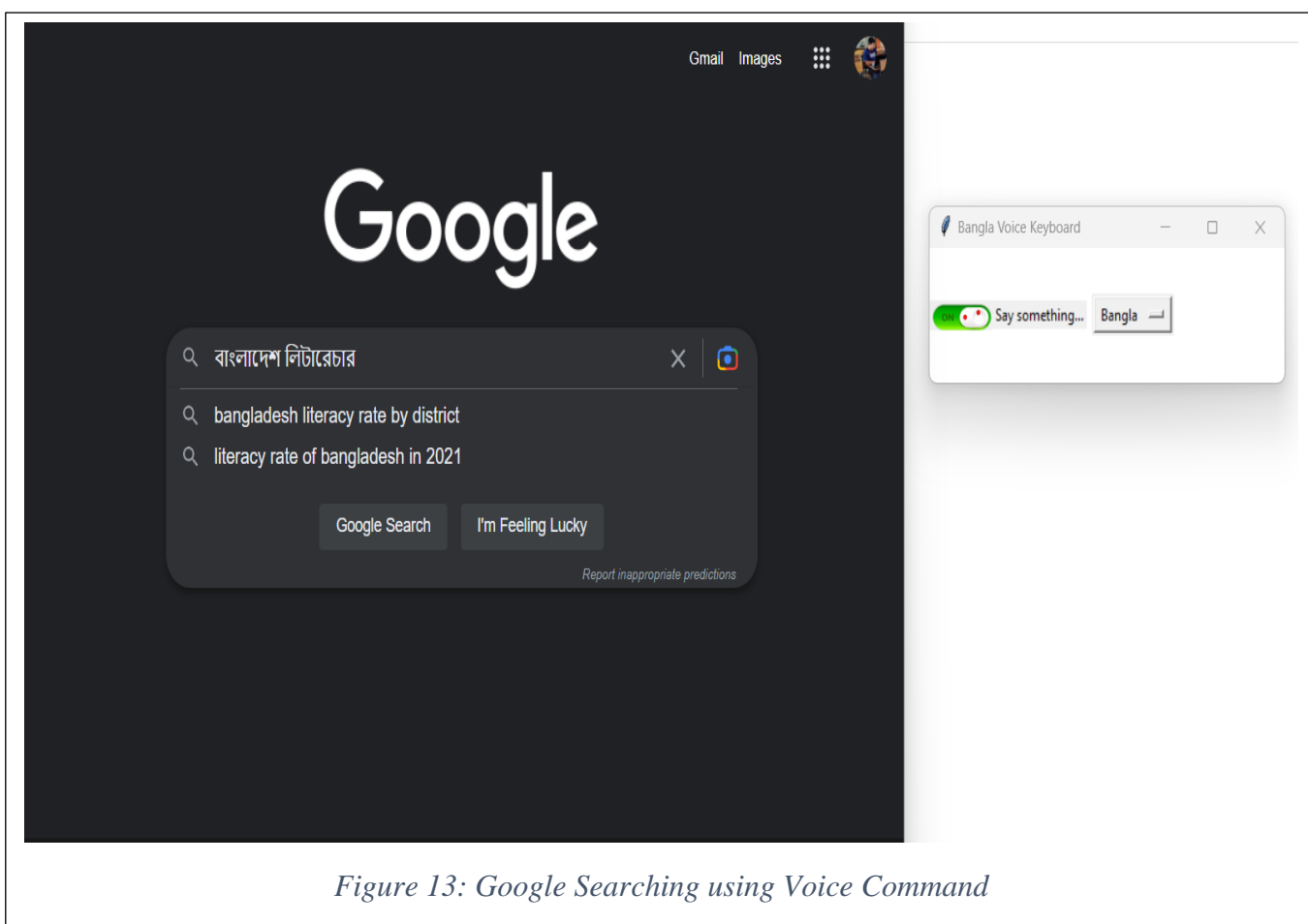


Figure 11: Content searching in YouTube





*Figure 13: Google Searching using Voice Command*



# Chapter 6

## Conclusions and Future Works

### 6.1 Conclusions

A fast input method is essential for modern times. Specially a voice keyboard which empowers people by offering the ability to write in both English and a native language of preference. The main objective of this work was to ensure accurate Bangla output in Unicode with the help of Google Cloud Speech API. This work produces satisfactory results in terms of producing the output in Unicode. The problem of using both Unicode and Bijoy at the same platform was tried to solve using Unicode (UTF-8) to Bijoy (ASCII Encoded) character mapping as a one to one mapping scheme of time complexity  $O(1)$ . This mapping was quite fast.

### 6.2 Future Works

However slight problem in mapping a particular complex letter exists. It is assumed that the problem is due to some technical characteristics of Bijoy font. However overall result is satisfactory in this stage of work. It is concluded that our concept is justified and shows good results.

## References

- [1]“Bijoy Ekushe.” <https://bijoyekushe.net.bd/> (accessed Nov. 22, 2022).
- [2]“Proshika Shabda. Get the software safely and easily.,” *Software Informer*. <https://proshika-shabda.software.informer.com/> (accessed Nov. 22, 2022).
- [3]“PRABARTAN 2000. Get the software safely and easily.,” *Software Informer*. <https://prabartan-2000.software.informer.com/> (accessed Nov. 22, 2022).
- [4]“Keyboard Layouts – Ekushey.” <https://ekushey.org/keyboards/> (accessed Nov. 22, 2022).
- [5]“Avro Keyboard - Unicode and ANSI compliant Free Bangla Typing Software and Bangla Spell Checker.” <https://www.omicronlab.com/avro-keyboard.html> (accessed Nov. 22, 2022).
- [6]“Download Free Font Munir.” <https://www.wfonts.com/font/munir> (accessed Nov. 23, 2022).
- [7]M. S. Sarkar, “Avro Phonetic Bangla Letters list অত্র ফনেটিক বাংলা,” *কিভাবে.কম*, Sep. 20, 2020. <https://kivabe.com/eng/avro-phonetic-bangla-letters-list/> (accessed Nov. 23, 2022).
- [8]“Bengali.” <https://unicode-table.com/en/blocks/bengali/> (accessed Nov. 22, 2022).
- [9]“UniBijoy keyboard layout download for Avro Keyboard,” *Shaharia’s Blog*, May 18, 2011. <https://blog.shaharia.com/unibijoy-keyboard-layout-in-avro-keyboard> (accessed Nov. 23, 2022).
- [10] “Bangla Unicode Font - Free Download.” <https://www.freebanglafont.com/catetory.php?b=173> (accessed Nov. 23, 2022).
- [11] S. Hasan, G. Rabbi, R. Islam, H. I. Bijoy, and A. Hakim, “Bangla Font Recognition using Transfer Learning Method,” presented at

- the 2022 International Conference on Inventive Computation Technologies (ICICT), 2022, pp. 57–62.
- [12] “KickKeys - Free download and software reviews - CNET Download.” [https://download.cnet.com/KickKeys/3000-2279\\_4-10057217.html](https://download.cnet.com/KickKeys/3000-2279_4-10057217.html) (accessed Nov. 23, 2022).
- [13] “12 Best Dictation Software 2022 [Voice To Text Software],” *Software Testing Help*. <https://www.softwaretestinghelp.com/best-dictation-software/> (accessed Nov. 23, 2022).
- [14] “Bangla Software - Banglapedia.” [https://en.banglapedia.org/index.php/Bangla\\_Software](https://en.banglapedia.org/index.php/Bangla_Software) (accessed Nov. 22, 2022).
- [15] “What is ASCII (American Standard Code for Information Interchange)?,” *WhatIs.com*. <https://www.techtarget.com/whatis/definition/ASCII-American-Standard-Code-for-Information-Interchange> (accessed Nov. 23, 2022).

