THAPATHALI CAMPUS
Institute of Engineering

A PROJECT REPORT

ON

INTERACTIVE DICTIONARY & PARAGRAPH AUTOCORRECT

IN PARTIAL FULLFILLMENT OF THE REQUIREMENTS FOR THE COURSE OF CT 401 COMPUTER PROGRAMMING

BACHELOR OF ELECTRONICS, COMMUNICATION AND INFORMATION ENGINEERING

SUBMITTED BY

ANUPAM BHATTARAI [THA076BEI008]

DIPU DAHAL [THA076BEI045]

JAGADISH SHRESTHA [THA076BEI010]

SUBMITTED TO

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

THAPATHALI CAMPUS

MARCH, 2020

This page is intentionally left blank.

DECLARATION

We hereby declare that the project work report entitled "Interactive Dictionary & Paragraph Autocorrect" submitted for the partial fulfillment of the requirements for the course of CT 401 Computer Programming is our original work and the Project Work Report has not formed the basis for the award of any degree, diploma, or other similar titles.

Signature

Name of Student : Anupam Bhattarai

Signature

Name of Student : Dipu Dahal

Signature

Name of Student : Jagadish Shrestha

ACKNOWLEDGEMENTS

Firstly, we would like to express our deep gratitude to Er. Saroj Shakya for letting us to do this wonderful project and providing guidelines. We would like to say thank you to the Electronics and Computer department of Thapathali Campus for providing us a place to learn and a space to create.

We would like to give special thanks to our classmates who encouraged, discussed and exchanged ideas of each other. Without their help, this project could not be imagined. Also, we would like to thank Github, Udemy, StackOverflow and other few websites for providing a good platform for learning and knowledge sharing.

ABSTRACT

Language is the primary form of communication and also hard to learn due to the presence of variety of words and rules. Likewise, it's tedious for us to remember every word meaning and set of rules while communicating through English language. So, as a part of our C programming project, we present "**Interactive Dictionary and Paragraph Autocorrect**" to reduce the complexity of English language.

This project report consists of detailed description about interactive dictionary and paragraph autocorrect along with their useful features. 'Paragraph Autocorrect' helps us to correct most of the mistakes while writing bunch of text documents or simply a paragraph and makes our writings more precise. On the other hand, 'Interactive Dictionary' helps us to find word meaning of every English word through graphical user interface and also consists of text-to-speech converter. The combined use of both of these features will help us a lot during verbal and written communication.

Key Words:

Interactive Dictionary; Paragraph Autocorrect; text-to-speech; graphical user interface.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE **PAGE**

LIST OF CHARTS

CHART **PAGE**

CHAPTER 1: INTRODUCTION

1.1 Background

We wanted to make a problem-solving project which would be helpful in our daily life, fun to code and somewhat unique. Then, we choose “Interactive Dictionary & Paragraph Auto-correct” which would help us increase our efficiency in English language and also improve our programming knowledge and skill.

1.2 Problem Statement

We feel trouble whenever we encounter a new word. Dictionaries (books) are not possible to be carried all the time. Even if we managed to carry them, we will have difficulty and it will also consume a lot of time, in finding words due to the bulkiness of the book. Also, there could be some small errors or spelling mistakes in our text document or digital writings. A highly mobile, interactive and a simple program may be the solution to this. In this project report, we have attempted to present a project for a similar purpose.

The program contains an easy interface with graphics to have the program as interactive as possible. Various libraries and header files have been used as an attempt to give user a nice experience of the program. The bugs have been minimized as much as possible and the program has been made to be as user friendly as possible.

1.3 Objectives

I. To design an algorithm to detect flaws in a paragraph or text file and also autocorrect those flaws.
II. To handle a dictionary file smartly for the detection of word and its meaning
III. To implement graphical user interface in the program for the enhancement of user experience.

IV. To implement text-to-speech feature in the dictionary.

1.4 Applications

This program has following applications:

1. ‘Paragraph Autocorrect’ can be used to detect and solve mistakes in large text documents which if manually edited would take a lot of time and effort.
2. Meaning of every English word could be found in less time and interactive way.

3. It will be useful for non-native English speakers to have the language understood properly.

1.5 Project Features

The major features of our project are as follows:

- Easy search and generation of meaning of most of the English words.
- Text-to-speech feature to be helpful for people with visual disability.
- Interactive interface for opening and saving of text file in ‘Paragraph Autocorrect’.
- Correction of spelling, case and formatting error in a text document.

1.6 Feasibility Analysis

1.6.1 Economic Feasibility

The program is simple and cheap, it requires low memory space and a compiler, which is available free on web.

1.6.2 Technical Feasibility

The program is easy to understand and technically easy to modify and update.

1.6.3 Operational Feasibility

The program can be operated in any computer with Windows Operating System which has access to c compiler.

1.7 System Requirement

1.7.1 Software Requirement

To run this program, a 32, 64-bit or higher Windows operating system is required. To edit the source code and compile it, IDE such as Codeblocks, Dev C++, Turbo C++, etc. is needed. As it is a simple GUI application in C language, it doesn't require much RAM.

1.7.2 Hardware Requirement

No any hardware except a computer or any electronic device which can run .exe files is required to run the GUI application formed after compilation.

CHAPTER 2: LITERATURE REVIEW

The early modern period was an era of great change for the English language. According to the OED's record, the number of words available to speakers of English more than doubled between 1500 and 1650. Many of the new words were borrowed into English from the Latin or Greek of the Renaissance (for example, hypotenuse), or from the far-off countries visited by travelers and traders (e.g. pangolin), and must have seemed hard to understand to many of the population. So, the concept of dictionary arose from there.

2.1 How did this idea come?

During the early modern period, there were significant demographic shifts in Britain towards an urbanized culture based in the big cities, such as London: the population of London increased eightfold over these years. In retrospect, one can argue the growing availability of books and other printed matter as the period developed—alongside the emergence of the grammar school as a focus for education (especially for boys)—meant that the scene was set for the emergence of the English dictionary.

2.2 Where and when did it start?

The first book generally regarded as the first English dictionary was written by Robert Cawdrey, a schoolmaster and former Church of England clergyman. In 1604 Cawdrey made use of wordlists published earlier in educational texts, such as Richard Mulcaster's 'Elementary (1582)' and Edmund Coote's 'The English Schoole-Master (1596)'. (The First Dictionaries of English) [1]

The first electronic dictionaries with interfaces designed for human users were an offshoot of calculator and PDA technology, and became available in 1978. These were the LK3000 produced by the Lexicon Corporation, Florida (the rights were acquired by Nixdorf (now Siemens) in 1979), the Craig M100 produced by the Craig Corporation, Japan, and Speak & Spell, an educational toy produced by Texas Instruments. The LK-3000, also known as the Lexicon, was designed in 1976 and patented in 1979 as an 'electronic dictionary and language interpreters.

In the early days of the internet the transfer was from older print dictionaries to the web, nowadays publishers turn to the internet for language data to inform new dictionaries in book form. Dictionary entries look less like dictionary (book) on the pages, and lexicographers (those who compile a dictionary) use non-static display functions such as the 'three-dimension search', where

related dictionary entries are graphically depicted in a kind of web of words spreading out from a central item. (Dictionaries in Electronic form) [2]

2.3 Advantages of Interactive Dictionary & Paragraph Autocorrect

- Faster result
- Trustable
- Convenient
- Faster correction of word file
- Easy and interactive GUI

2.4 Why to use program like ours?

Our application is just a prototype of electronic dictionary and autocorrect feature with few functions. But advanced applications which have similar concepts can be used to implement more auto-correct and dictionary functions with faster speed and convenience.

CHAPTER 3: DESIGN & METHODOLOGY

In this program, we have used basically two programs (excluding some little GUIs), the dictionary and the paragraph autocorrect program. The dictionary helps us to find the right meaning of the given word while paragraph autocorrect looks and solves the error in spelling, case and format of the text document. These two programs are described below:

3.1 Interactive Dictionary

The dictionary program takes input from the user in 'search' box. When a word is typed and 'Generate' button is pressed, this program searches the word in data.json file and displays its corresponding meaning if the word is found else displays 'word not found' message. When text-to-speech is turned 'on', the word along with its meaning is spoken out.

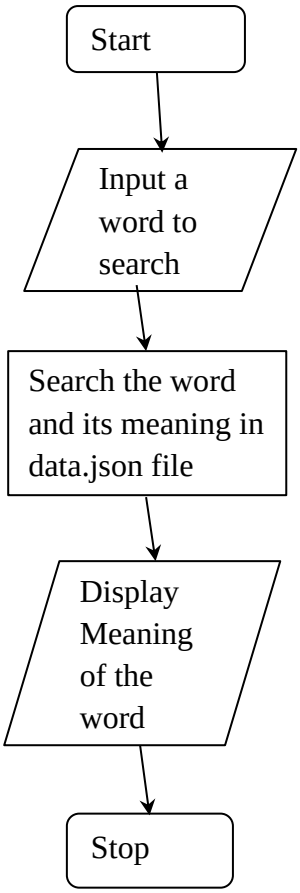

Fig 3.1 Block diagram for dictionary program

3.2 Paragraph Auto-correct

'Paragraph Auto-correct' deals with spelling error, lowercase or uppercase error, extra or less character and formatting error in a text document. First of all, the user has to open a text file using 'open file' button or manually typing the text. Now, when the 'Auto-correct' button is pressed, the error in given text document is corrected. User can save the corrected text in a text file for future use. Vertical-scroll feature is also implemented in the input box of 'Paragraph Auto-correct' for large text file.

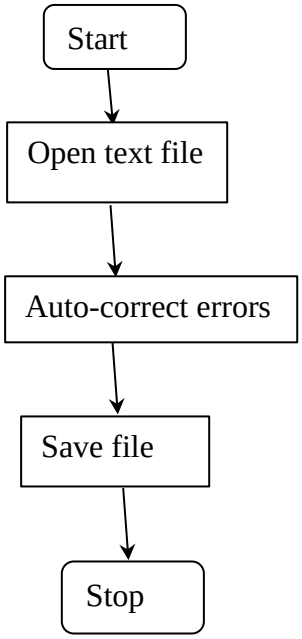

Fig 3.2 Block diagram for Paragraph Auto-correct

CHAPTER 4: IMPLEMENTATION AND RESULT

4.1 Implementation Detail

Many functions, structures and arrays have been used in the program. Some of the important functions and labels are as follows:

4.1.1 “data.json” file:

The “data.json” file has the collection of most of the English words and their meaning. This file is used to find word-meaning in dictionary program.

4.1.2 Structures

Structure named ‘spellStore’ is used to store the word with spelling mistake and its position to alert the user for spelling mistake in the ‘Paragraph Auto-correct’ program.

4.1.3 dictionary() function

The dictionary function is used to retrieve the inputted word in ‘search’ box and search in the ‘data.json’ file for its meaning. It also sends the word and its meaning to text-to-speech if it is ‘on’.

4.1.4 Auto_Correct() function

The Auto_Correct function is used to correct the common mistakes in the text present in input box of ‘Paragraph Auto-correct’. It calls the other four sub-functions for the correction of extra character, unformatted text, improper case and spelling.

4.1.5 Other functions

Some other important functions such as tts() which is used for text-to-speech, open_file() and save_file() for opening and saving file in ‘Paragraph Auto-Correct’ are also implemented.

4.2 Result Analysis

The dictionary application provides meaning of the searched word and comes with the following layout:

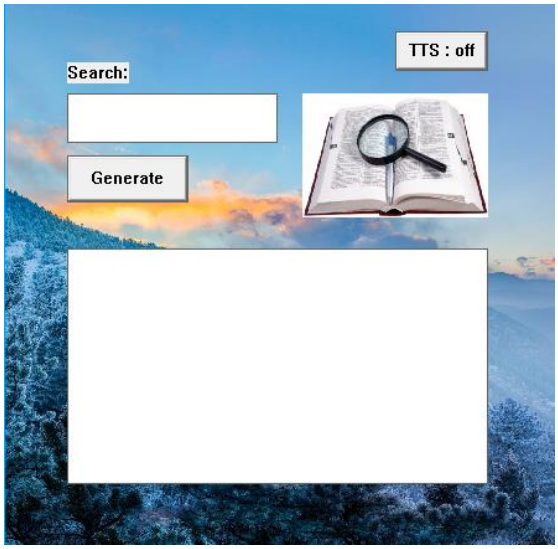

Fig 4.1 Layout of dictionary app

The ‘Paragraph Auto-correct’ app has the following layout:

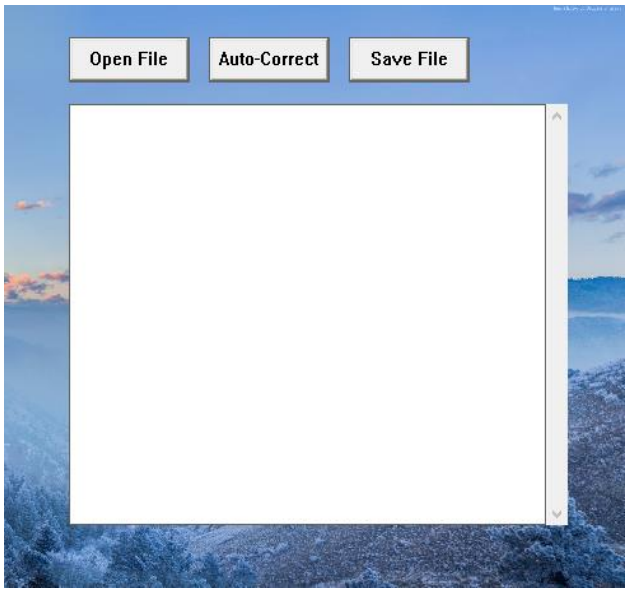

Fig. 4.2 Layout of Paragraph Auto-correct app

As, in the first picture above, the dictionary app has a search box to input the word, generate button to find the meaning of the inputted word and a big output box to display the meaning of the inputted word. There is also a ‘tts’ button to turn text-to-speech ‘on’ and ‘off’ manually if needed.

The second picture shows ‘Paragraph Autocorrect’ app with three button and a big output box with vertical scroll enabled. ‘Open file’ and ‘Save File’ button is used to open and save the text file respectively whereas ‘Auto-correct’ button is used to correct errors in the text file. If there is one or more spelling mistakes in the text file, then a pop-up dialog box opens to show the spelling mistake and an input box to check for the correct word.

CHAPTER 5: CONCLUSION & FURTHER WORK

5.1 Conclusion

Hence, we designed an "Interactive Dictionary & Paragraph Auto-correct" as a part of our C programming project of first semester.

The program is made with simple user interface and is also contained with text-to-speech converter which is supposed to be helpful for people with visual disability and also a paragraph auto-correction which can access the files, correct them for spelling, case and spacing errors and save them for future use.

The resources for GUI programming using C language are limited and it's hard to find a good tutorial. So, we have done the best we can to implement various features in the program and to make it useful and interactive.

The program was completed in the time frame of 10 weeks as below and might require further works to make a much effective program.

S. N.	WORKING SCHEDULE	Start: 8 Dec.,2016									No. of Weeks
		1	2	3	4	5	6	7	8	9	10
1	Project Identification										
2	Proposal Writing										
3	Interactive Dictionary										
4	Paragraph Autocorrect										
5	GUI										
6	Testing and Debugging										

5.1: Gantt chart

5.2 Further Works

Due to the time constraints and complexity of program, we could only design our program to a limited level. However, there is always room for improvement.

We have used <windows.h> library for the GUI & provide a proper and simple experience of the program. However, this was our first exposure to the GUI and with deeper knowledge about the GUI, we could have made the program much simple and convenient to the user. For more precision & effectiveness in auto-correction, the grammar correction can also be added to the program.

The use of windows API has made the program limited to Windows Operating System; using other libraries could have made the program more flexible and convenient for other operating systems.

Also keep in mind that the project had to be completed in limited time between the pressure of studies and exams. With more time, the program could have been made more efficient and error free. Bugs might show up when the program is being used more and more, which can be debugged whenever they show up.

REFERENCES

[1] J. Simpson, "Oxford English Dictionary- Blog," [Online]. Available: https://public.oed.com/blog/the-first-dictionaries-of-english/.

[2] H. Nesi, " Dictionaries in Electronic form," in The Oxford History of English Lexicography, Oxford, Oxford University Press, 2008, pp. 455-478.

APPENDICES

- **Function used for dictionary:**

```
void dictionary(HWND hWnd)
{
   char name[30],name2[30],search[32],out[1000],out2[1000],out3[1000];
   int i=0,flag=1,flag2=1,cL=430,val;

   GetWindowText(hName,name,30);
   GetWindowText(hName,out,500);
   strcpy(name2,name);
   strlwr(name);

   if(strcmp(name,"")==0)
   {
      val = MessageBoxW(hWnd,L"You did not enter anything
!",NULL,MB_RETRYCANCEL | MB_ICONERROR);
   }

   switch(val)
   {
      case IDCANCEL:
         DestroyWindow(hWnd);
         break;

      case IDRETRY:
         return;
   }

   if(strcmp(name,"")==0)
      return;

   FILE *file = fopen("data.json","rb");

   fseek(file,0,SEEK_END);
   int size = ftell(file);
   rewind(file);

   char *str = malloc(size+1);
   fread(str,size,1,file);
   str[size] = '\0';
```

```
x:
strcpy(search,"\"");
strcat(search,name);
strcat(search,"\"");

char *ptr = strstr(str,search);

if(ptr==NULL)
{
   if(strcmp(name,name2)==0)
   {
      if(name[0]>='a' && name[0] <='z' && flag!=0)
      {
         name[0] = name[0] - 32;
         flag=0;
         goto x;
      }
      else if(flag2!=0)
      {
         strupr(name);
         flag2=0;
         goto x;
      }
      strcpy(out2,"Word Not Found. Please enter the correct word.");
      strcpy(out3,out2);
   }
   else
   {
      strcpy(name,name2);
      goto x;
   }
}
else
{
   ptr += strlen(search)+3;
   int co=0;
   while(*(ptr)!=']')
   {
      if(*ptr=='[')
         out[i] = ' ';
      else if(*ptr=='\"')
         out[i] = ' ';
```

```
        else
            out[i] = *ptr;

        if(*ptr=='\"')
        {
            if(co%2==0 && (*(ptr-2)== ',' || *(ptr-1)== '['))
                out2[i] = '#';
            else
                out2[i] = ' ';

            co++;
        }
        else if(*ptr==',' && out2[i-1]==' ')
        {
            out2[i-1] = 13;
            out2[i] = '\n';
        }
        else
        {
            out2[i] = *ptr;
        }

        ptr++;
        i++;
    }
    out[i]='\0';
    out2[i]='\0';

    strcpy(out3,name);
    strcat(out3,":");
    strcat(out3,out);
}

free(str);
fclose(file);

SetWindowText(hOut,out2);

if(blindButton%2 == 1)
{
    if(strlen(out3)>cL)
    {
        strncpy(out,out3,cL);
```

```
            out[cL] = '\0';
        }
        else
        {
            strcpy(out,out3);
        }
        tts(out);
    }
}
```

For any queries, contact us at:

dipu234dahal@gmail.com