

**Visvesvaraya Technological University
Belagavi-590018, Karnataka**



A Mini Project Report on

“Document Processing”

Submitted in partial fulfillment of the requirement for the
File Structure Laboratory with Mini-Project
[17ISL68]

**Bachelor of Engineering
in
Information Science and Engineering**

Submitted by
SWATHI BIRADAR[1JT17IS046]

Under the Guidance of
Mr.Vadiraja A
Asst.Professor,Department of ISE



**Department of Information Science and Engineering
Jyothy Institute of Technology
Tataguni, Bengaluru-560082**

Jyothy Institute of Technology
Department of Information Science and Engineering
Tataguni, Bengaluru-560082



CERTIFICATE

Certified that the mini project work entitled **“DOCUMENT PROCESSING”** carried out by **SWATHI BIRADAR [1JT17IS046]** bonafide student of Jyothy Institute Of Technology, in partial fulfillment for the award of **Bachelor of Engineering in Information Science and Engineering** department of the **Visvesvaraya Technological University, Belagavi** during the year **2019-2020**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

Prof. Vadiraja A
Guide, Asst, Professor
Dept. Of ISE

Dr. Harshvardhan Tiwari
Associate Professor and HOD
Dept. OF ISE

External Viva Examiner :

Signature with Date:

- 1.
- 2.

ACKNOWLEDGMENT

Firstly, we are very grateful to this esteemed institution **Jyothy Institute of Technology** for providing us an opportunity to complete our project.

We express our sincere thanks to our Principal **Dr. Gopalakrishna K** for providing us with adequate facilities to undertake this project.

We would like to thank **Dr. Harshvardhan Tiwari, Associate Professor and Head of Information Science and Engineering** Department for providing for his valuable support.

We would like to thank our guide **Prof. Vadiraja A , Asst. Prof.** for his keen interest and guidance in preparing this work.

Finally, we would thank all our friends who have helped us directly or indirectly in this project.

SWATHI BIRADAR

1JT17IS046

ABSTRACT

The knowledge acquisition bottleneck has become the major impediment to the development and application of effective information systems. To remove this bottleneck, new document processing techniques must be introduced to automatically acquire knowledge from various types of documents. In this project I have tried implementing document processing using python language. Using this project one can insert text or number into document with ease and it even enables user to display full document to console or part of it by specifying the number of lines required by giving beginning and ending numbers. The user can also search for a string and get all occurrences of it with the line number where its present and can even find and replace any existing word with the new word, which will be reflected in the original file in matter of seconds. Overall this project enables features for user to insert ,display ,edit and search within the given document.

INDEX

SL NO	Description	Page No.
	Chapter 1	
1	Introduction	1 - 3
1.1	Introduction to File Structures	1
1.2	Introduction to File System	2
1.3	Introduction to Document Processing	3
	Chapter 2	
2	Requirement analysis and design	4 - 5
2.1	Domain Understanding	4
2.2	Classification of Requirements	4
2.3	Time Analysis	5
	Chapter 3	
3	Implementation	6 - 7
	Chapter 4	
4	Result and Analysis	8 - 9
	Chapter 5	
5	Observation and Conclusion	

CHAPTER 1

INTRODUCTION

CHAPTER 1

INTRODUCTION

1.1 Introduction to File Structures

In simple terms, a file is a collection of data stored on mass storage (e.g., disk or tape) .But there is one important distinction that must be made at the outset when discussing file structures. And that is the difference between the logical and physical organization of the data.

On the whole a file structure will specify the logical structure of the data, that is the relationships that will exist between data items independently of the way in which these relationships may actually be realized within any computer. It is this logical aspect that we will concentrate on. The physical organization is much more concerned with optimizing the use of the storage medium when a particular logical structure is stored on, or in it. Typically for every unit of physical store there will be a number of units of the logical structure (probably records) to be stored in it.

For example, if we were to store a tree structure on a magnetic disk, the physical organization would be concerned with the best way of packing the nodes of the tree on the disk given the access characteristics of the disk.

Like all subjects in computer science the terminology of file structures has evolved higgledy-piggledy without much concern for consistency, ambiguity, or whether it was possible to make the kind of distinctions that were important.

It was only much later that the need for a well-defined, unambiguous language to describe file structures became apparent. In particular, there arose a need to communicate ideas about file structures without getting bogged down by hardware considerations.

1.2 Introduction to File System

In computing, a file system or file system controls how data is stored and retrieved. Without a file system, information placed in a storage medium would be one large body of data with no way to tell where one piece of information stops and the next begins. By separating the data into pieces and giving each piece a name, the information is easily isolated and identified. Taking its name from the way paper-based information systems are named, each group of data is called a “file”. The structure and logic rules used to manage the groups of information and their names is called a “file system”.

There are many different kinds of file systems. Each one has different structure and logic, properties of speed, flexibility, security, size and more. Some file systems have been designed to be used for specific applications. For example, the ISO 9660 file system is designed specifically for optical discs.

File systems can be used on numerous different types of storage devices that use different kinds of media. The most common storage device in use today is a hard disk drive. Other kinds of media that are used include flash memory, magnetic tapes, and optical discs. In some cases, such as with tmpfs, the computer's main memory (random-access memory, RAM) is used to create a temporary file system for short-term use.

Some file systems are used on local data storage devices; others provide file access via a network protocol (for example, NFS, SMB, or 9P clients). Some file systems are “virtual”. Meaning that the supplied “files” (called virtual files) are computed on request (e.g. procfs) or are merely a mapping into a different file system used as a backing store. The file system manages access to both the content of files and the metadata about those files. It is responsible for arranging storage space; reliability, efficiency, and tuning with regard to the physical storage medium are important design considerations

1.3 Document Processing

Documents serve to archive and communicate information. Document processing is the activity of operating on information captured in some form of persistent medium. Humans, endowed with the capacity to read, write, and think, are the principal actors in document processing. The invention of the modern digital computer, supported by various key technologies, has revolutionized document processing. Because information can be coded in other media that is read and written by the computer—from punched cards in the early 1960s to magnetic tapes , disks, and optical CDs (compact discs) today—it is not always necessary for documents to be on paper for processing.

Document processing can be implemented by any programming language and can be used to generate required results. When there are more and more large, complex documents people want to be able to process it quicker and efficiently like searching a particular string or finding and replacing all strings in a document . At times like this , we use programs or applications to achieve it by ease.

CHAPTER 2

REQUIREMENT ANALYSIS AND

DESIGN

CHAPTER 2

REQUIREMENT ANALYSIS AND DESIGN

2.1 Domain Understanding

Document processing can be implemented by any programming language and can be used to generate required results.

2.2 Classification of Requirements

User Requirements

OS : Windows / Linux / MAC

Software and Hardware Requirements

Programming Languages: Python

RAM: 1GB

2.3 System and Time Analysis

As soon as the user runs the program, he will be prompted to enter the file name to process it in the console of editor. Then he will be provided with a menu to choose how and what process has to be done with the input text file. Based on the input given by the user the respective functions are called to produce suitable expected output.

Time analysis is "An ordered sequence of values of a variable at equally spaced Time intervals. It is used to understand the determining factors and structure behind the observation data, choose a model to forecast, thereby leading to better decision making."

Below in the table we have taken search, replace and insert time, so we can analyze time taken to process the code. Which is discussed in detail in chapter 4

Word	Avg. Search time	Avg. Replace time	Avg. Insert time
String	1.9 sec	4.2 sec	2.2 sec
Numbers	2.1 sec	3.9 sec	2 sec
Combination of both	2.4 sec	7.2 sec	2.4 sec

As we can see in the above table rows represent the type of data that was modified and columns indicate the average time taken for each of the operation to be completed successfully.

CHAPTER 3

IMPLEMENTATION

CHAPTER 3

IMPLEMENTATION

Introduction

This project involves a single python program which is used for document processing. It consists of seven main functions namely:

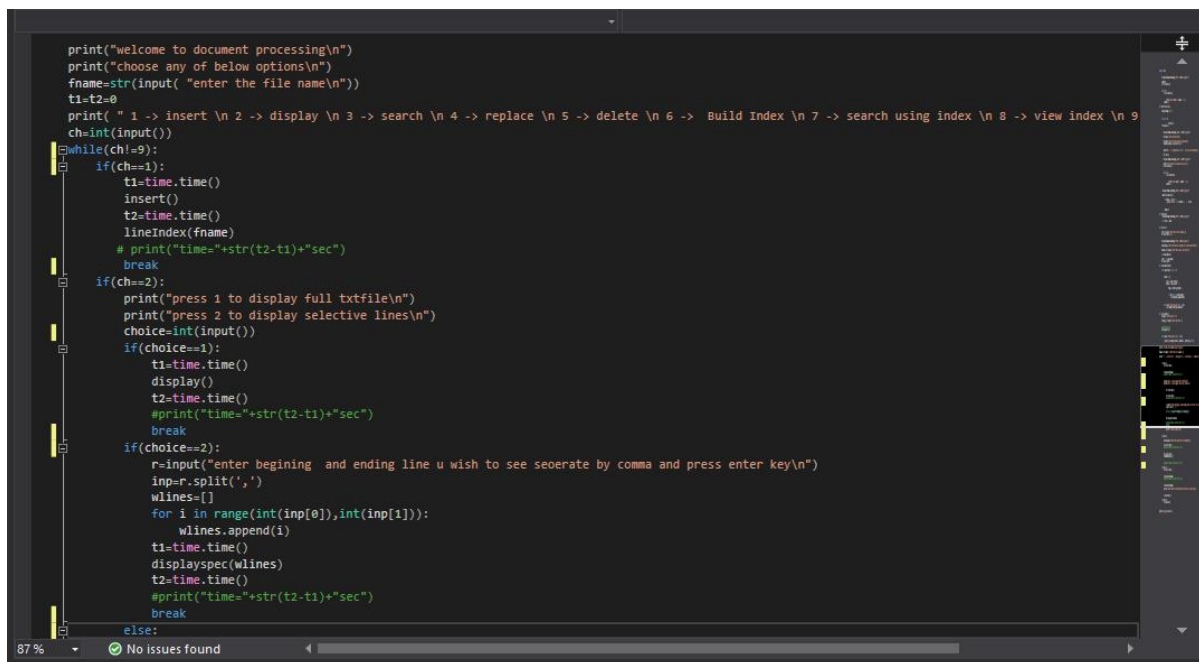
1. Insert
2. Search
3. Replace
4. Display
5. Display specific
6. Delete.
7. Quit

we will be going through each function how it has been implemented in this project. Apart from these functions it has driver code which is main part of this project.

Driver Code:

In below fig 3.1 and 3.2 we have implemented the driver code in python which consists of 6 options representing different function. It is a menu driven program where, I've implemented it using while and if statements based on input given by the user each of the six functions are called by this module of the project.

At the start of the program user is asked to enter the name of the file he wishes to modify note that file should be present in the same directory. Then he will be displayed the menu to choose what need's to be done as the user enters the input he will b driven to the function which he asked for later after the execution of function the control is returned back to driver program where user can choose another function from menu and go on with it ,then when all modifications done user can simply quit the program by entering 6 which is for quit().

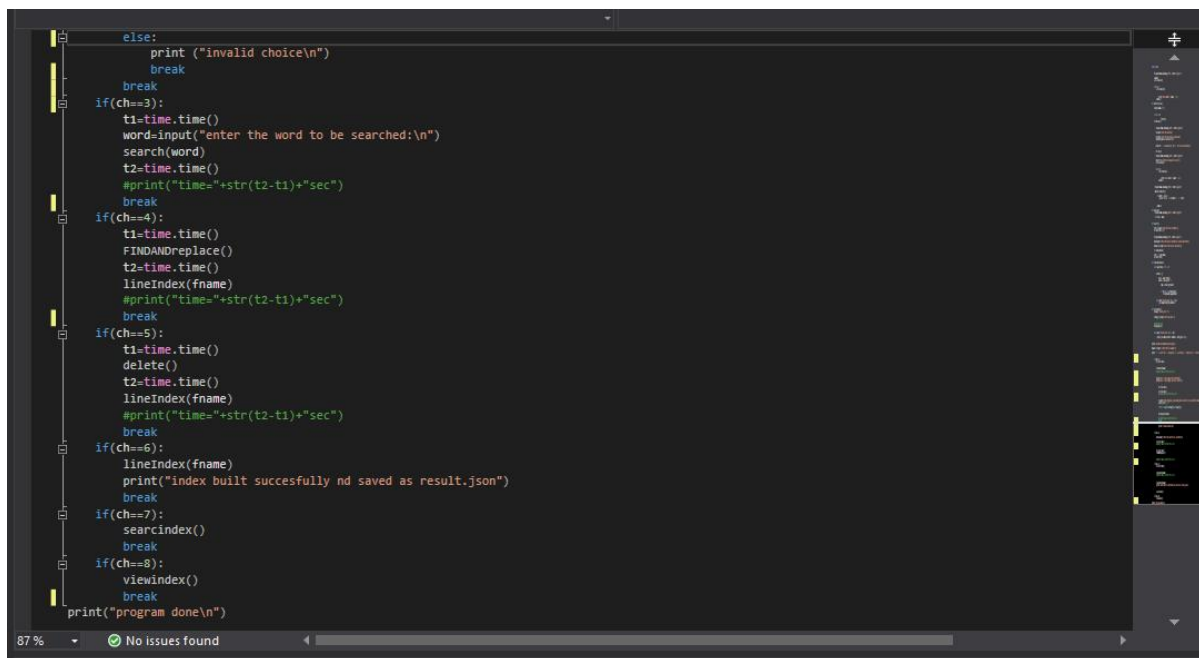


```

print("welcome to document processing\n")
print("choose any of below options\n")
fname=str(input( "enter the file name\n"))
t1=t2=0
print( " 1 -> insert \n 2 -> display \n 3 -> search \n 4 -> replace \n 5 -> delete \n 6 -> Build Index \n 7 -> search using index \n 8 -> view index \n 9 -> exit\n")
ch=int(input())
while(ch!=9):
    if(ch==1):
        t1=time.time()
        insert()
        t2=time.time()
        lineIndex(fname)
        # print("time="+str(t2-t1)+"sec")
        break
    if(ch==2):
        print("press 1 to display full txtfile\n")
        print("press 2 to display selective lines\n")
        choice=int(input())
        if(choice==1):
            t1=time.time()
            display()
            t2=time.time()
            #print("time="+str(t2-t1)+"sec")
            break
        if(choice==2):
            r=input("enter beginning and ending line u wish to see seerate by comma and press enter key\n")
            inp=r.split(',')
            wlines=[]
            for i in range(int(inp[0]),int(inp[1])):
                wlines.append(i)
            t1=time.time()
            displayspec(wlines)
            t2=time.time()
            #print("time="+str(t2-t1)+"sec")
            break
    else:
        print("invalid choice\n")
        break

```

Fig 3.1



```

else:
    print ("invalid choice\n")
    break
    break
if(ch==3):
    t1=time.time()
    word=input("enter the word to be searched:\n")
    search(word)
    t2=time.time()
    #print("time="+str(t2-t1)+"sec")
    break
if(ch==4):
    t1=time.time()
    FINDANDreplace()
    t2=time.time()
    lineIndex(fname)
    #print("time="+str(t2-t1)+"sec")
    break
if(ch==5):
    t1=time.time()
    delete()
    t2=time.time()
    lineIndex(fname)
    #print("time="+str(t2-t1)+"sec")
    break
if(ch==6):
    lineIndex(fname)
    print("index built succesfully nd saved as result.json")
    break
if(ch==7):
    searchindex()
    break
if(ch==8):
    viewindex()
    break
print("program done\n")

```

Fig 3.2

Insert()

The insert method does not return anything. It only updates the current file. we simply take input from console and then write it to the file. below is the implementation of the insert function in this project

I've opened the file in append mode and then read the string or data that is to be inserted to the file into a string variable 'txt' and then its written at the end of the file and control I returned back to the calling program

Display()

This functions displays whole file from beginning line till the end it take no parameter, below is the snippet of how its being implemented in this project.

I've opened the file in read text mode and then copied all the information present inside a file to a variable name data which is then iterated and whole file is printed line by line in the console

Displayspec()

Where as the display function prints the whole file, Display can be used to display particular lines from the file it takes a parameter called wlines which is a list containing the line number user asked for below is the fig how this was implemented in this project

I've opened the file in read text mode and then initialized count variable to 1 which I helping to print the lines we are just going to iterate line by line and compare count with the range user has given if the count is inside the range then the line I sprinted else go to next iteration till u reach the end of the file

Search()

Search function is used to search the specified word and displays the line number's wherever the word is present . we just compare the string with each line and print the line number and string, as the string from document matches to the string taken as input. Below is the fig showing implementation of the same .

This function doesn't take any parameter ,I've simply opened the file in read text mode then copies whole data of the file to a variable "s". then each line is split into words based on spaces and iterated word by word and line by line comparing the the word from file with user given word when they match line number and line is printed on console this repeats until the whole file is searched

Find&replace()

This function takes no parameter, it is achieved by using built in function replace which takes two parameters namely the original string and string which is to be replaced and return true if successful. Below is fig showing how its implemented in this project

I've opened the file in text mode and den read all the data to a variable named data and prompt user for word which is to be replaced and word replacing then we replace the words by using built function replace and store it to a another variable and write it to the file by opening it in write mode and close file .

Delete()

Delete function which delete the word specified in the file. This function takes no parameter it deletes the specific line given by user it makes use of search to help user to visualize the lines he needs to delete and the enter the line number .

I've opened the file in text mode and then read all the information present in the file to a variable called data. Ask user to enter keyword or sentence which he needs to delete then displays all line numbers and occurrence's of that word asking user to enter the line number of the line which he wants to delete and based on that line number the line is deleted and its written into the file by6 opening it in write mode and then close the file

Build Index()

This function builds index for each and every word present in the file and stores it to result.json. This function takes no parameter . I've opened the file in read mode and read all data to contents variable and dictionary name result is created in which key and value pairs are built for each word in the the file which c an be later used to search or delete based on index created. And new file result.json is created and the generated index is copied to it and then file is closed

View Index()

This function enables user to view the index built and saved in result.json on the console. It takes no parameter , the files opened in read mode and whole content is copied to a variable and then print line by line to console

Search Using Index()

This function enables user to search the file with help of keyword in result.json .The result .json is opened in read mode and it takes the input string and extracts all the values associated with certain keyword and all the values are the line numbers which will be passed to displayspec() function to print all the lines on the console. It takes no parameter .

Quit()

This is a built in function to terminate the execution of the program after using it.

CHAPTER 4

RESULT AND ANALYSIS

CHAPTER 4

Result And Analysis

Result

```
welcome to document processing

choose any of below options

enter the file name
script.txt
1 -> insert
2 -> display
3 -> search
4 -> replace
5 -> delete
6 -> Build Index
7 -> search using index
8 -> view index
9 -> quit
1
enter the line to add
hai i like weka
program done

Press any key to continue . . .
```

Fig 4.1

In fig 4.1 we have selected the first option which is insert this simply takes the input given from the console and writes in on to the file.

```
welcome to document processing

choose any of below options

enter the file name
script.txt
1 -> insert
2 -> display
3 -> search
4 -> replace
5 -> delete
6 -> Build Index
7 -> search using index
8 -> view index
9 -> quit

press 1 to display full txtfile

press 2 to display selective lines

1
=====

-----

README

-----
```

Fig 4.2

In fig 4.2 we have selected option second which is display function . further we can choose either we have to select and display the whole program or the specific lines. Hence we have to select option 1 for full display or option 2 for spec display fig 4.4 shows the result when option 2 is selected that is specific lines from 10,15 and fig 4.3 displays whole file .

```
11. Copyright:
-----

The core WEKA system is distributed under the GNU public
license. Please read the file COPYING.

Packages may be distributed under various licenses - check the
description of the package in question for license details.

-----

hai i like weka

program done

Press any key to continue . . .
print("program done\n")
```

Fig 4.3

```
1 -> insert
2 -> display
3 -> search
4 -> replace
5 -> delete
6 -> Build Index
7 -> search using index
8 -> view index
9 -> quit
2
press 1 to display full txtfile
press 2 to display selective lines
2
enter beginning and ending line u wish to see seerate by comma and press enter key
10,15
lineno: 10 ->
lineno: 11 ->      Copyright (C) 1998-2019 University of Waikato
lineno: 12 ->
lineno: 13 ->      web: http://www.cs.waikato.ac.nz/~ml/weka
lineno: 14 ->
program done
Press any key to continue
```

Fig 4.4

```

welcome to document processing
choose any of below options

enter the file name
script.txt
1 -> insert
2 -> display
3 -> search
4 -> replace
5 -> delete
6 -> Build Index
7 -> search using index
8 -> view index
9 -> quit
3
enter the word to be searched:
hai
['-----']
line number: 435 : hai i like weka

program done

Press any key to continue . . .

```

Fig 4.5

In fig 4.5 we have selected option 3 which is search function. This function search the word from the program and displays the the word or sentence and the line number.

```

enter the file name
script.txt
1 -> insert
2 -> display
3 -> search
4 -> replace
5 -> delete
6 -> Build Index
7 -> search using index
8 -> view index
9 -> quit
4
enter the word:
hai
['-----']
line number: 435 : hai i like weka

enter the word to be replaced
hello
hai is replaced by hello in file succesfully

the word to be replaced lie in:
['-----']
line number: 435 : hello i like weka

program done

Press any key to continue . . .

```

Fig 4.6

In fig 4.6 we have selected 4 option which is replace. Firstly we search for the word which as to be replaced in the file and then specify the word to be replaced with and display it

```

Welcome to document processing
Choose any of below options
Enter the file name
script.txt
1 -> insert
2 -> display
3 -> search
4 -> replace
5 -> delete
6 -> Build Index
7 -> search using index
8 -> view index
9 -> quit
5
Enter the word or sentence to search and deletehello
['=====']
line number: 435 : hello i like weka

Enter the line no to delete435
program done

Press any key to continue . . .

```

Fig 4.7

Fig 4.7 and fig 4.8 shows the deletion operation the user searches for the keyword and displays all the matching lines where user can select a line number and it gets deleted in the file

```

413 If you find any bugs, send a bug report to the wekalist mailing list.
414
415 1) For core Weka components, i.e. everything in the main weka.jar file
416 (not including packages), send a bug report to the wekalist mailing
417 list.
418
419 2) For packages, check the package description (either online at
420 Sourceforge or by using the package management system) and contact the
421 maintainer of the package directly.
422
423 -----
424
425 11. Copyright:
426 -----
427
428 The core WEKA system is distributed under the GNU public
429 license. Please read the file COPYING.
430
431 Packages may be distributed under various licenses - check the
432 description of the package in question for license details.
433
434 -----
435
436
437

```

Fig 4.8

```

welcome to document processing

choose any of below options

enter the file name
script.txt
1 -> insert
2 -> display
3 -> search
4 -> replace
5 -> delete
6 -> Build Index
7 -> search using index
8 -> view index
9 -> quit
6
index built succesfully nd saved as result.json
program done

Press any key to continue . . .

```

Fig 4.9

In above fig shows that line index was built and saved in result.json which will be further used to search and modify the data

```

welcome to document processing

choose any of below options

enter the file name
script.txt
1 -> insert
2 -> display
3 -> search
4 -> replace
5 -> delete
6 -> Build Index
7 -> search using index
8 -> view index
9 -> quit
7
enter the word distributed
lineno: 427 -> The core WEKA system is distributed under the GNU public

lineno: 430 -> Packages may be distributed under various licenses - check the

program done

Press any key to continue . . .

```

Fig 4.10

In fig 4.10 I have selected the 7th option search by index which finds the given word in the file and retrieves the word long with the line number and displays it.


```

welcome to document processing

choose any of below options

enter the file name
script.txt
1 -> insert
2 -> display
3 -> search
4 -> replace
5 -> delete
6 -> Build Index
7 -> search using index
8 -> view index
9 -> quit
8
{
  "": [
    1,
    2,
    3,
    4,
    5,
    6,
    7,
    8,
    9,
    10,
    11,
    12,

```

Fig 4.11

In fig 4.11 and 4.12 shows that view index option read the index generated and saved in result.json and display them in console.

```

274,
276,
336,
341,
356,
360,
393,
394,
412
],
"your": [
  57,
  69,
  174,
  178,
  179,
  187,
  279,
  283,
  356,
  396,
  399
],
"{first_value,": [
  146
]
}
program done
Press any key to continue . . . _
time.time()

```

Fig 4.12

CHAPTER 5

Observation and Conclusion

Observation



Fig 5.1

It has been observed that time taken for data processing increases with the increase in the size of the file. We have performed the different commands such as insert, display, display spec, search, replace & delete with various size files varying from 500, 1000, 5000, 10000 & 15000 and for four to five times and have taken average all. Hence the larger the data the slower the process.

Conclusion

We have successfully implemented Document Processing which helps us in administrating the data present in the file and for managing the tasks performed in the file by using the python language . We have also used function called as indexing which makes it less time consuming since it retrieves the specified word with the line number. It is faster and easy to perform. Finally, we have have a program where the user can not only read but also can write and edit accordingly with ease.

References

1. https://books.google.co.in/books?id=_oeoANtvHTcC&pg=PA71&dq=line+index+python&hl=en&sa=X&ved=0ahUKEwj15f_fnebpAhW5xjgGHYTdBNAQ6AEIGDAC
2. <https://youtu.be/dVJ2x3UAsGQ>
3. https://en.m.wikipedia.org/wiki/Text_processing
4. https://en.m.wikipedia.org/wiki/Document_processing
5. https://www.w3schools.com/python/python_json.asp
6. <https://docs.python.org/3/library/json.html>