**INDEX**

**SERIAL NO. CONTENT PAGE NO.**

1 Introduction 2

2 Software Requirement Specification 3

3 Design 4

4 Implementation 6

5 Testing And Results 11

7 Conclusion 20

8 References 21

**1. INTRODUCTION**

**1.1 Problem Definition:**

We often need to save different soils and be able to access it when required. This can be achieved by storing the soils in files which can be easily transferred. **Soil Predictor** is a desktop application that aims to provide a solution by predicting specific crop type by taking the soil information and retrieving them in a specific format.

**1.2 Concepts of File Structures used:**

**Techniques Description**

Indexing Basic file structure used for the project. Involves creation of an index file and a data file. The index file contains the data file record’s primary key and the byte offset location in the data file used carrying out operation effectively.

Add Adding a variable length record into a file without letting it affect the previous content of the file.

+9

Search Using indexing, search for any record with any field data in a particular record entered (starting with or part of a name or phone number).

Modify Using indexing and search operation, get the record that user wished for and update any changes required on existing fields of the records.

Sorting Depending on user’s wish sort the data in the index files on increasing name or phone numbers in the file. A new search index file is created in the process for reducing the burden of sorting the records again from sort. All changes for sort are made in the search index file.

**2. SOFTWARE REQUIREMENT SPECIFICATION**

**2.1 Functional Requirements**

The main functional requirements of **Soil Predictor** are:

**ID Description**

FR1 Be able to create a record which contains soil details like type, location, pH, etc..

FR2 To store these records in a file.

FR3 To access the records based on a primary key.

FR4 Ability to search result in a list view.

FR5 If no matches are found, the user must be informed clearly. FR8 Filtering results based on combination of all attributes.

**2.2 Non-Functional Requirements**

The features that we would like the application to have apart from the functional requirements are listed below.

**ID Description**

QR1 Ability to handle unexpected situations like wrong input in place of the number.

QR2 A timely response to the input(Response Time) i.e. the fastness to the search a record, measurements obtained from 1000 searches during testing o more than 2 seconds 100% of the time.

**2.3 Hardware & Software Requirements**

The hardware and software requirements for PhoneBook applications are:

**ID Description**

QR3 Hard Drive Space, application’s need for hard drive space for storing files running the application effectively. No more than 15 Bytes

QR4 A desktop with any operating system that supports Python. Tkinter Basic file operations must be supported.

**3. DESIGN**

**3.1 SYSTEM ARCHITECTURE**

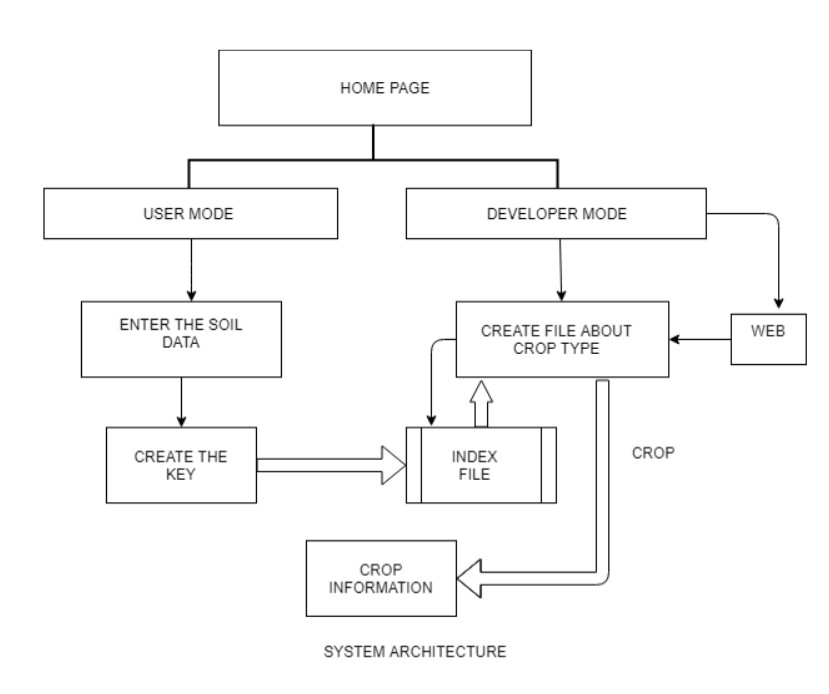


Fig 3.1

**3.2 Basic Data Flow Diagram**

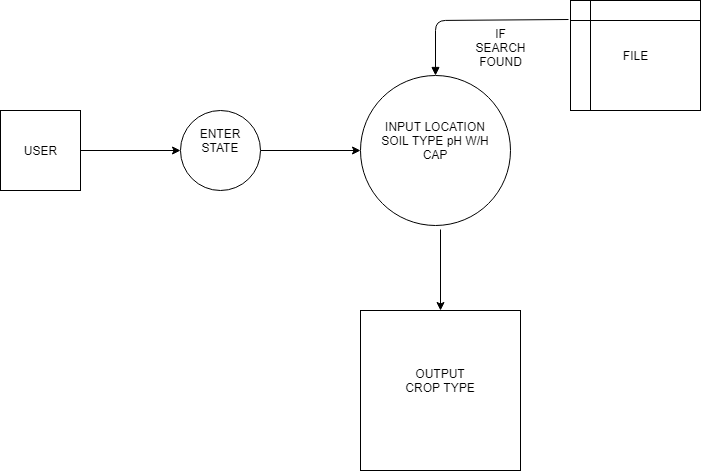
USER DATA FLOW DIAGRAM****

Fig 3.2a

DEVELOPER DATA FLOW DIAGRAM

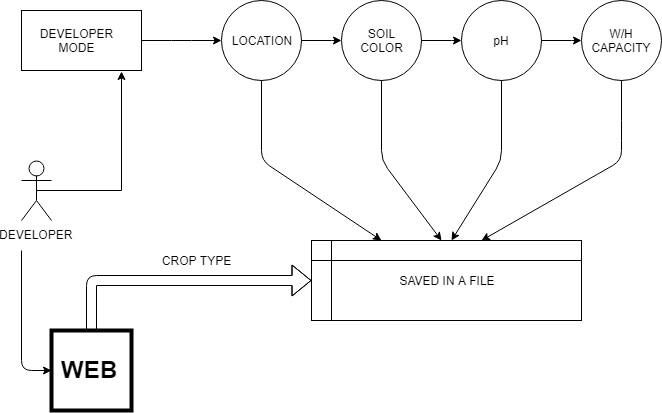
****

Fig 3.2b

**4. IMPLEMENTATION**

**4.1 Softwares Used**

The programming languages used in this application was Python 3. The entire application was developed on Spyder and Anaconda. The reason for selecting Python 3 was that it serves wide variety of purpose. So, it allows this application to grow.

The implementation part of this project started by first stating the requirements. Once the requirements were clear, the user interface was designed. All possible interactions and operations of the application were planned. After all these tasks were completed, the coding part commenced with the development of the user interface.

The UI was designed in Python using Tkinter library. Tkinter is a software platform for creating and delivering desktop applications, as well as rich internet applications that can run across a wide variety of devices.

Simultaneously along with the development of the UI, the backend was developed and later integrated with the UI and tested if it satisfied all the functional requirements.

**4.2 Functionalities**

**4.2.1 Add Record:**

The code for adding a record to the data file and to the index files is given below.

import random

#pid=1000

typesoil=['Alluvial','Red','Black','Arid','Laterite']

location=['U.P.','Maharashtra','Rajasthan','Uttarakhand','Karnataka']

ph=[3,5,7,9]

whcapacity=[40,50,60,70]

#croptype=['A','B','C','D','E']

#p=1

#pid=str(i+1)+str(j+1)+str(k+1)+str(l+1)

croptype='A'

f=open('mainfilenew.txt','w')

for i in range(len(typesoil)):

for j in range(len(location)):

for k in range(len(ph)):

for l in range(len(whcapacity)):

pid=str(i+1)+str(j+1)+str(k+1)+str(l+1)

if pid[0]=='2' and pid[1]=='3':

croptype='D'

elif (pid[0]=='3' or pid[0]=='4' or pid[0]=='5') and pid[1]=='1':

croptype='D'

elif ( pid[0]=='1' or pid[0]=='4' or pid[0]=='5' ) and pid[1]=='5':

croptype='D'

elif (pid[0]=='2' or pid[0]=='3' or pid[0]=='4' or pid[0]=='5') and pid[1]=='4':

croptype='D'

elif (pid[0]=='1' or pid[0]=='4' or pid[0]=='5' ) and pid[1]=='2':

croptype='D'

cstring=str(pid)+'|'+str(typesoil[i])+'|'+str(location[j])+'|'+str(ph[k])+'|'+str(whcapacity[l])+'|'+random.choice(croptype)+"$"

croptype='a'

f.write(cstring)

f.write("\n")

#print(pid,'|',i,'|',j,'|',k,'|',l,end="$")

f.close()

**4.2.5 Searching a record:**

The method to search a record in the file is given below.

import re

f=open('mainfile.txt','r')

f2=open('index.txt','r')

'''

location=(input("Enter the location: \npress 1 for U.P.\npress 2 for for Maharashtra\npress 3 for Rajasthan \npress 4 for Uttrakhand\npress 5 for Karnataka "))

colortype=(input("Enter the Colortype: \npress 1 for Alluvial\npress 2 for Red\npress 3 for Black\npress 4 for Arid "))

whcapacity=(input("Enter the Water holding capacity: \npress 1 for 30-49%\npress 2 for capacity between 40-49%\npress 3 for capacity between 50-59%\npress 4 for capacity between 60-70%"))

ph=(input("Enter the ph: 1 for 3\n2 for 5\n3 for 7\n4 for 9"))

s='|'+colortype+'|'+location+'|'+ph+'|'+whcapacity+'|'

flag=0

for line in f:

if(s==line[4:13]):

print(line,"\n")

'''

location=(input("Enter the location: \npress 1 for U.P.\npress 2 for for Maharashtra\npress 3 for Rajasthan \npress 4 for Uttrakhand\npress 5 for Karnataka "))

colortype=(input("Enter the Colortype: \npress 1 for Alluvial\npress 2 for Red\npress 3 for Black\npress 4 for Arid "))

whcapacity=(input("Enter the Water holding capacity: \npress 1 for 30-49%\npress 2 for capacity between 40-49%\npress 3 for capacity between 50-59%\npress 4 for capacity between 60-70%"))

ph=(input("Enter the ph: 1 for 3\n2 for 5\n3 for 7\n4 for 9"))

s2=colortype+location+ph+whcapacity

flag2=0

for line2 in f2:

if s2==line2[0:4]:

offs=int(line2[5:])

print(offs)

f.seek(offs,0)

print(f.readline()[-3])

**GUI FILE**

from tkinter import \*

from tkinter import messagebox

def printfun():

soilval=soil.get()

locval=location1.get()

phval=pH.get()

wh=whcap.get()

for i in range(len(typesoil)):

if soilval==typesoil[i]:

soilval=str(i+1)

break

for i in range(len(location)):

if locval==location[i]:

locval=str(i+1)

break

for i in range(len(ph)):

if phval==ph[i]:

phval=str(i+1)

break

for i in range(len(whcapacity)):

if wh==whcapacity[i]:

wh=str(i+1)

break

print(soilval,locval,phval,wh)

f=open('mainfile.txt','r')

f2=open('index.txt','r')

s2=soilval+locval+phval+wh

flag2=0

for line2 in f2:

if s2==line2[0:4]:

offs=int(line2[5:])

print(offs)

f.seek(offs,0)

z=f.readline()[-4:-2]

print(z)

# messagebox.showinfo("yes",z)

f.close()

f2.close()

datalist=[]

so=open("datasoil.txt",'r')

x=''

line=so.readline()

while line:

datalist.append(line)

line=so.readline()

print (datalist)

index=-1

for i in range(len(datalist)):

if(datalist[i][:2]==z):

index=i

if index==-1:

x="No Data Found"

else:

for i in range(index+1,len(datalist)):

if(datalist[i]=="\*\*\*\n"):

break

x+=datalist[i]

print(x)

new=Toplevel(root)

new.geometry("700x2000")

lt=Label(new,text="Hello "+e1.get()+" Preferred crop in your area :\n "+x)

lt.place(x=15,y=70)

mainloop()

root=Tk()

root.title("Crop Predictor")

root.geometry("500x500")

heading=Label(text="Crop Prediction",bg="LightSteelBlue1",fg="black",width="500",height="3")

heading.pack()

###################################################################################

typesoil=['Alluvial','Red','Black','Arid','Laterite']

location=['U.P.','Maharashtra','Rajasthan','Uttarkhand','Karnataka']

ph=[3,5,7,9]

whcapacity=[40,50,60,70]

##################################################

slt=Label(text="Soil Type")

slt.place(x=15,y=70)

lct=Label(text="Location")

lct.place(x=250,y=70)

phs=Label(text="pH value")

phs.place(x=15,y=200)

wc=Label(text="W.C. capacity")

wc.place(x=250,y=200)

soil=StringVar()

soilch=OptionMenu(root,soil,'Alluvial','Red','Black','Arid','Laterite')

soilch.config(font=("Arial",10))

soilch.place(x=100,y=70)

location1=StringVar()

locch=OptionMenu(root,location1,'U.P.','Maharashtra','Rajasthan','Uttarkhand','Karnataka')

locch.config(font=("Arial",10))

locch.place(x=350,y=70)

pH=IntVar()

phch=OptionMenu(root,pH,3,5,7,9)

phch.config(font=("Arial",10))

phch.place(x=100,y=200)

whcap=IntVar()

whch=OptionMenu(root,whcap,40,50,60,70)

whch.config(font=("Arial",10))

whch.place(x=350,y=200)

name=Label(text="Name")

name.place(x=100,y=300)

e1 = Entry(root)

e1.place(x=200, y=300)

search=Button(text="Search",width="30",height="2",command=printfun,bg="LightSteelBlue1")

search.place(x=120,y=350)

root.mainloop()

**5. TESTING & RESULTS**

**5.1 TESTING OBJECTIVES**

 Functional Testing: Black box testing to ensure coverage of all functionality mentioned in the SRS. In particular, functions tested include:

 Addition of record (can be verified by viewing data file)

 Searching a record (will be visible in the user interface)

 Defect Testing: To root out any bugs and ensure the system works as expected. In particular to ensure that the addition, modification, and deletion of the records work as intended without affecting other records.

**5.2 TESTING PROCEDURES USED Black box testing:**

**Equivalence class testing**

Set of all test cases was partitioned into mutually disjoint subsets whose union is the entire set and one test case from each subset was chosen. There are two important implications for testing:

 The fact that the entire set is represented provides a form of completeness

 The dis jointness assures a form of non-redundancy

The equivalence classes decided upon for the variable “Soil Attributes” were

 set of valid location, color, pH, Water Holding Capacity

 set of invalid location, color, pH, Water Holding Capacity.

**5.3 RESULTS**

**5.3.1 Home Page:**

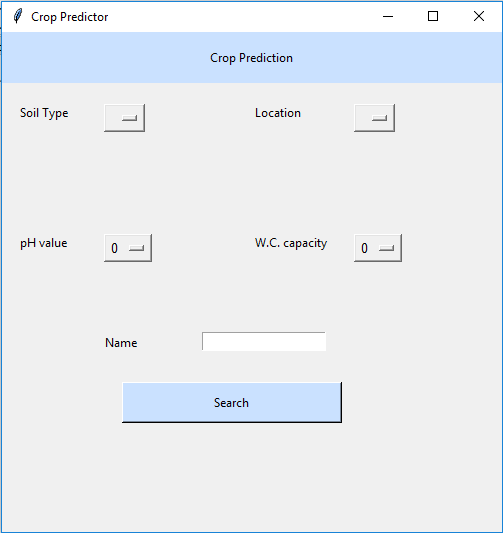


Fig 5.3.1

**5.3.2 Filled Entries Page:**

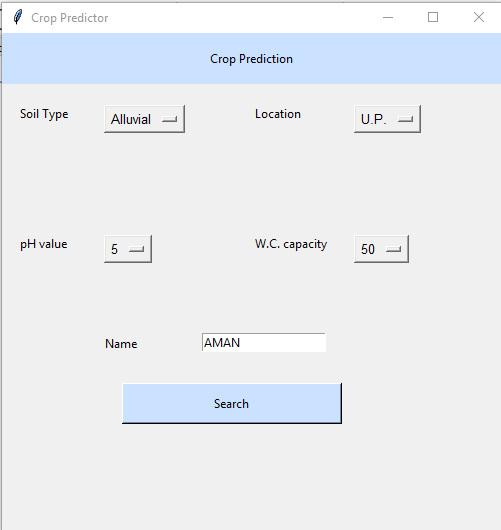


Fig 5.3.2

**5.3.3. Display / Search Records :**

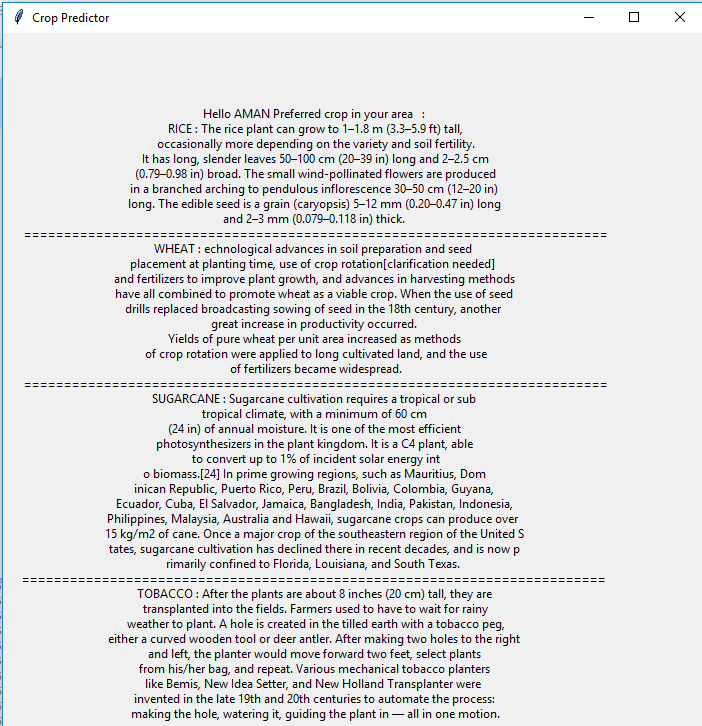


Fig 5.3.3

**5.3.4. Edit / Search Records :**

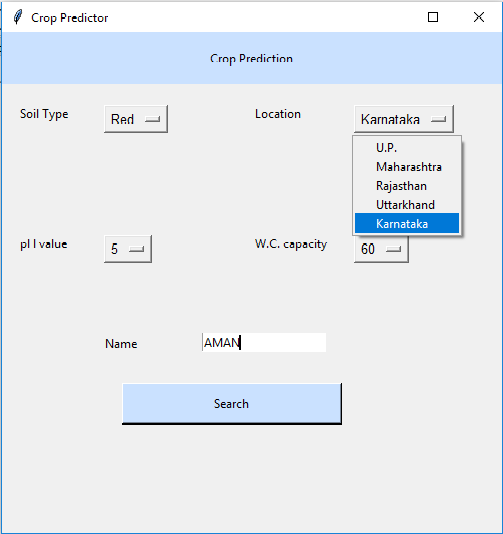
****

Fig 5.3.4

**5.3.5. Edit / Search Records Results :**

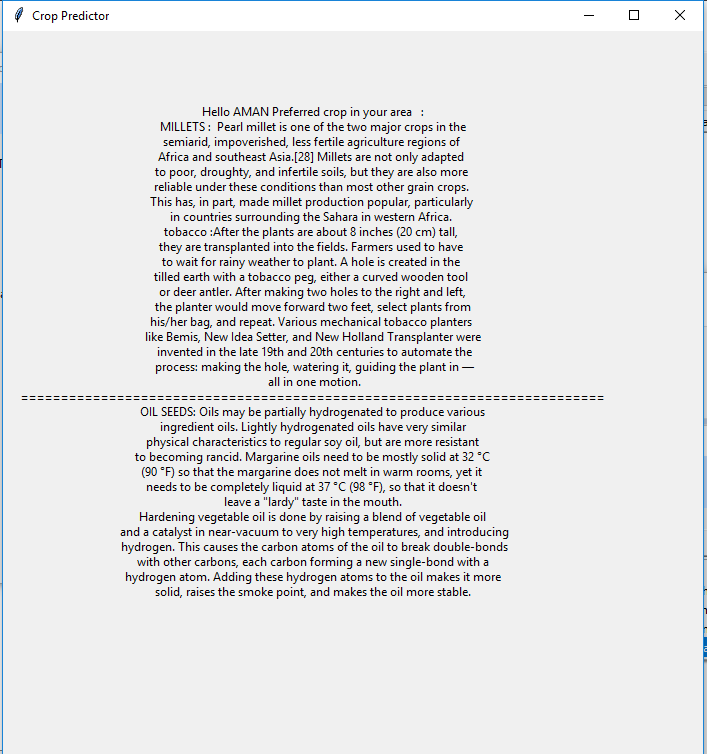
****

Fig 5.3.5

**5.3.6. Soil Data With Crop Match :**

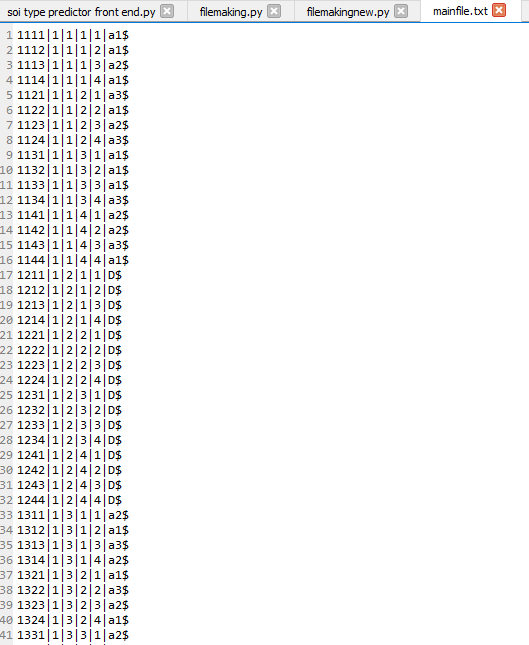
****

Fig 5.3.6

**5.3.7. Index File :**

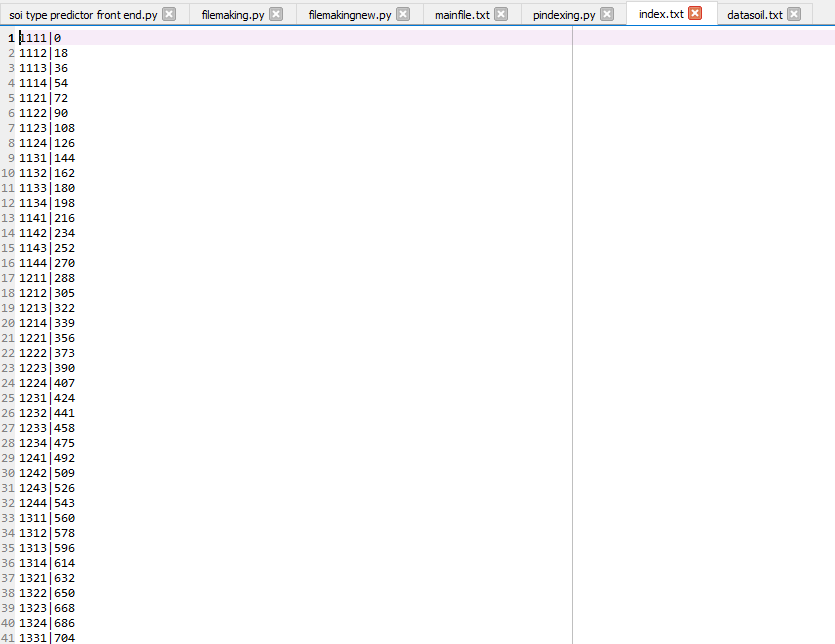
****

Fig 5.3.7

**5.3.7. Crop Data**

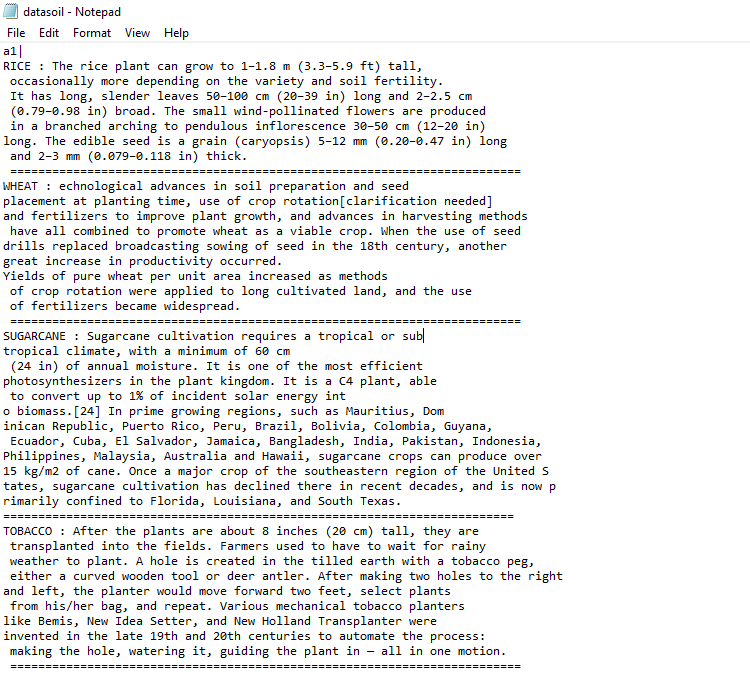
****

Fig 5.3.8

**6. CONCLUSION**

The application Crop Predictor has successfully demonstrated the functional requirements by performing the required operations as intended. It performs addition of the records, searching, and prediction of Crops without affecting other records.

Future work planned included:

 Predict a Crop when no data is available in data set, i.e. data predicting techniques.

 Adding more States and attributes of soil.

All in all, the Crop Predictor developed supports all the primitive operations and models the real world well.

**REFERENCES**

1. File Structures: An Object-Oriented Approach With C++ :Michael J.Folk, Bill Zoellick, Greg Riccardi;

2. K.R. Venugopal, K.G. Srinivas, P.M. Krishnaraj: File Structures Using C++, Tata McGraw-Hill, 2008.

3. Scot Robert Ladd: C++ Components and Algorithms, BPB Publications, 1993.

4. Raghu Ramakrishan and Johannes Gehrke: Database Management Systems, 3rd Edition, McGraw Hill, 2003.

5. [Wikipedia](http://www.learncpp.com/)

6. Crop Info

7. Almanac.com

8. AgriFarming.com