Mini Project

On

Directory Crawler

(CredB)

Name of the Students

2173038	Anant Shahi
2173017	Aditya Srivastava
2173056	Atrayee Chatterjee
2173040	Aniruddh D Sarkar

Project Guide

Prof. Nagesh Jadhav

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MIT-ADT UNIVERSITY, SCHOOL OF ENGINEERING, RAJBAUG, LONI, INDIA

MIT School of Engineering

Department of Computer Science & Engineering

RAJBAUG, LONI, INDIA

CERTIFICATE

This is to certify that the Mini Project entitled '**Directory Crawler(CredB)',** submitted by **Anant Shahi, Atrayee Chatterjee, Aditya Srivastava and Aniruddh D Sarkar** is a record of bonafide work carried out by themin the partial fulfillment of the requirement for the award of Degree of B Tech (CSE) at MIT School of Engineering, Rajbaug, Loni, Pune under the MIT ADT University. This work is done during the year 2018-19, Semester III.

Date: ……………….

--	---
Prof. Nagesh Jadhav	Dr. Rajneeshkaur Sachdeo-Bedi
Project Guide	Dean Engineering,
MIT School of Engineering	Head, Department of CSE
Loni Kalbhor, Pune	MIT School of Engineering Loni-Kalbhor, Pune

Acknowledgement

Apart from our efforts, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

We would like to show our greatest appreciation to Prof. Nagesh Jadhav. We can’t say thank you enough for his tremendous support and help. We felt motivated and encouraged every time we attended his classes. Without his encouragement and guidance, this project would not have been materialized.

The guidance and support received from all the members who contributed and who are contributing to this project, was vital for the success of the project. We are grateful for their constant support and help.

Name of students:

Anant Shahi

Atrayee Chatterjee

Aditya Srivastava

Aniruddh Sarkar

Abstract

A crawler is a program that visits Web sites and reads their pages and other information in order to create entries for a search engine index.Crawlers are typically programmed to visit sites that have been submitted by their owners as new or updated. Entire sites or specific pages can be selectively visited and indexed. Crawlers apparently gained the name because they crawl through a site a page at a time, following the links to other pages on the site until all pages have been read.

In our computer labs, during the time of practical examinations, one of the lab members has to manually go through each and every workstation and check for any file that can be maliciously used during the times of practical. This process is very time consuming. To overcome this scenario and to reduce human efforts, we have developed a directory crawler, which will crawl all the directories of every workstation, have all the file details which are present in the workstation compiled in a single file. End user will be provided with GUI using which all the files as well any single file or piece of information can be retrieved on a single click.

This offline search engine, the name we like to call it, has it's roots in python. This crawler employs PHP and MySQL on different instances.

Table of Contents

Introduction

A crawler is a program that visits Web sites and reads their pages and other information in order to create entries for a search engine index. Crawlers are typically programmed to visit sites that have been submitted by their owners as new or updated. Entire sites or specific pages can be selectively visited and indexed. Crawlers apparently gained the name because they crawl through a site a page at a time, following the links to other pages on the site until all pages have been read.

In our computer labs, during the time of practical examinations, one of the lab members has to manually go through each and every workstation and check for any file that can be maliciously used during the times of practical. This process is very time consuming. To overcome this scenario and to reduce human efforts, we have developed a directory crawler, which will crawl all the directories of every workstation, have all the file details which are present in the workstation compiled in a single file. End user will be provided with GUI using which all the files as well any single file or piece of information can be retrieved on a single click.

This offline search engine, the name we like to call it, has it's roots in python. This crawler employs PHP and MySQL on different instances.

Problem Definition

Write a modular program using python which can crawl the directories and files in workstations that are connected either wired or wirelessly to a system giving the required output with respect to input search keywords specified by end-user.

Features of Project

- CredB can crawl whole workstation(s) which are connected to the host system either wirelessly or with wired connection.
- It is equipped with GUI based front end which provides comfortable access to end user.
- CredB is real time offline crawler where any minor changes are also reflected in the results and databases are updated in real time.
- It is very stable
- It runs in the background and updates its databases with moderate disk and memory usage

Platform/Technology – Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code reliability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by metaprogramming and meta objects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.

Python's large standard library, commonly cited as one of its greatest strengths, provides tools suited to many tasks. For Internet-facing applications, many standard formats and protocols such as MIME and HTTP are supported. It includes modules for creating graphical user interfaces, connecting to relational databases, generating pseudorandom numbers, arithmetic with arbitrary precision decimals, manipulating regular expressions, and unit testing.

Some parts of the standard library are covered by specifications, but most modules are not. They are specified by their code, internal documentation, and test suites (if supplied). However, because most of the standard library is cross-platform Python code, only a few modules need altering or rewriting for variant implementations.

As of March 2018, the Python Package Index (PyPI), the official repository for third-party Python software, contains over 130,000 packages with a wide range of functionality, including:

- Graphical user interfaces
- Web frameworks
- Multimedia
- Databases
- Networking

- Test frameworks
- Automation
- Web scraping
- Documentation
- System administration
- Scientific computing
- Text processing
- Image processing

Most Python implementations (including CPython) include a read–eval–print loop (REPL), permitting them to function as a command line interpreter for which the user enters statements sequentially and receives results immediately.

Other shells, including IDLE and IPython, add further abilities such as auto-completion, session state retention and syntax highlighting.

As well as standard desktop integrated development environments, there are Web browser-based IDEs; SageMath (intended for developing science and math-related Python programs); PythonAnywhere, a browser-based IDE and hosting environment; and Canopy IDE, a commercial Python IDE emphasizing scientific computing.

Libraries that we have used in python

Import time:

This module in python helps to deal with time and date format in several different ways. Some of the functions of time modules are as follows:

Sr.No.	Function with Description
1	**time.altzone** The offset of the local DST timezone, in seconds west of UTC, if one is defined. This is negative if the local DST timezone is east of UTC (as in Western Europe, including the UK). Only use this if daylight is nonzero.
2	**time.asctime([tupletime])** Accepts a time-tuple and returns a readable 24-character string such as 'Tue Dec 11 18:07:14 2008'.
3	**time.clock( )** Returns the current CPU time as a floating-point number of seconds. To measure computational costs of different approaches, the value of time.clock is more useful than that of time.time().
4	**time.ctime([secs])** Like asctime(localtime(secs)) and without arguments is like asctime( )
5	**time.gmtime([secs])** Accepts an instant expressed in seconds since the epoch and returns a time-tuple t with the UTC time. Note : t.tm_isdst is always 0
6	**time.localtime([secs])** Accepts an instant expressed in seconds since the epoch and returns a time-tuple t with the local time (t.tm_isdst is 0 or 1, depending on whether DST applies to instant secs by local rules).
7	**time.mktime(tupletime)** Accepts an instant expressed as a time-tuple in local time and returns a floating-point value with the instant expressed in seconds since the epoch.

8	**time.sleep(secs)** Suspends the calling thread for secs seconds.
9	**time.strftime(fmt[,tupletime])** Accepts an instant expressed as a time-tuple in local time and returns a string representing the instant as specified by string fmt.
10	**time.strptime(str,fmt='%a %b %d %H:%M:%S %Y')** Parses str according to format string fmt and returns the instant in time-tuple format.
11	**time.time( )** Returns the current time instant, a floating-point number of seconds since the epoch.
12	**time.tzset()** Resets the time conversion rules used by the library routines. The environment variable TZ specifies how this is done.

Import OS

This module provides a portable way of using operating system dependent functionality.

Notes on the availability of these functions:

- The design of all built-in operating system dependent modules of Python is such that as long as the same functionality is available, it uses the same interface; for example, the function os.stat(path) returns stat information about *path* in the same format (which happens to have originated with the POSIX interface).
- Extensions peculiar to a particular operating system are also available through the os module, but using them is of course a threat to portability.
- All functions accepting path or file names accept both bytes and string objects, and result in an object of the same type, if a path or file name is returned.

Here some common useful functions of OS library are mentioned for better understanding

OS functions

import os

Executing a shell command

os.system()

Get the users environment

os.environ()

Returns the current working directory.

os.getcwd()

Return the real group id of the current process.

os.getgid()

Return the current process's user id.

os.getuid()

Returns the real process ID of the current process.

os.getpid()

Set the current numeric umask and return the previous umask.

os.umask(mask)

Return information identifying the current operating system.

os.uname()

Change the root directory of the current process to path.

os.chroot(path)

Return a list of the entries in the directory given by path.

os.listdir(path)

Create a directory named path with numeric mode mode.

os.mkdir(path)

Recursive directory creation function.

os.makedirs(path)

Remove (delete) the file path.

os.remove(path)

Remove directories recursively.

os.removedirs(path)

Rename the file or directory src to dst.

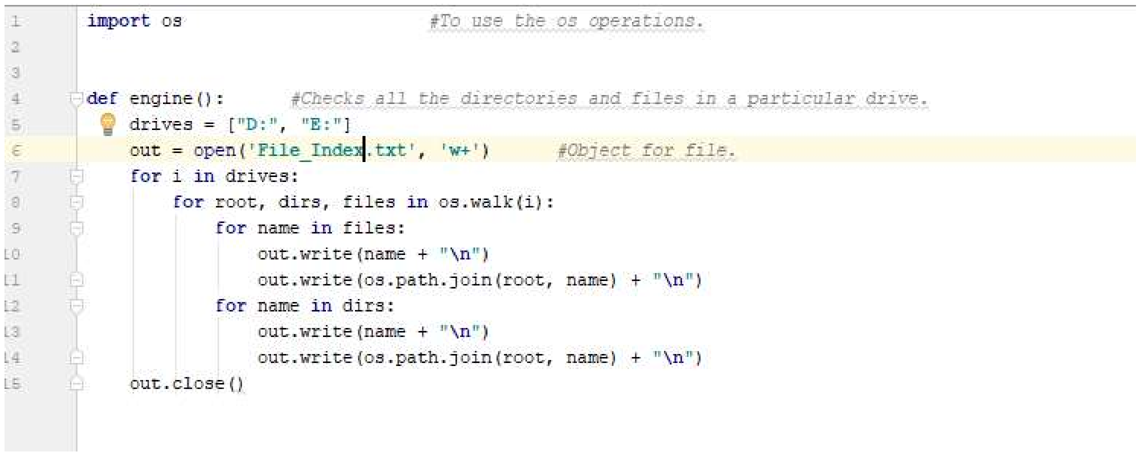
os.rename(src, dst)

Remove (delete) the directory path.

os.rmdir(path)

Flowchart and Module Explanation

Module 1:

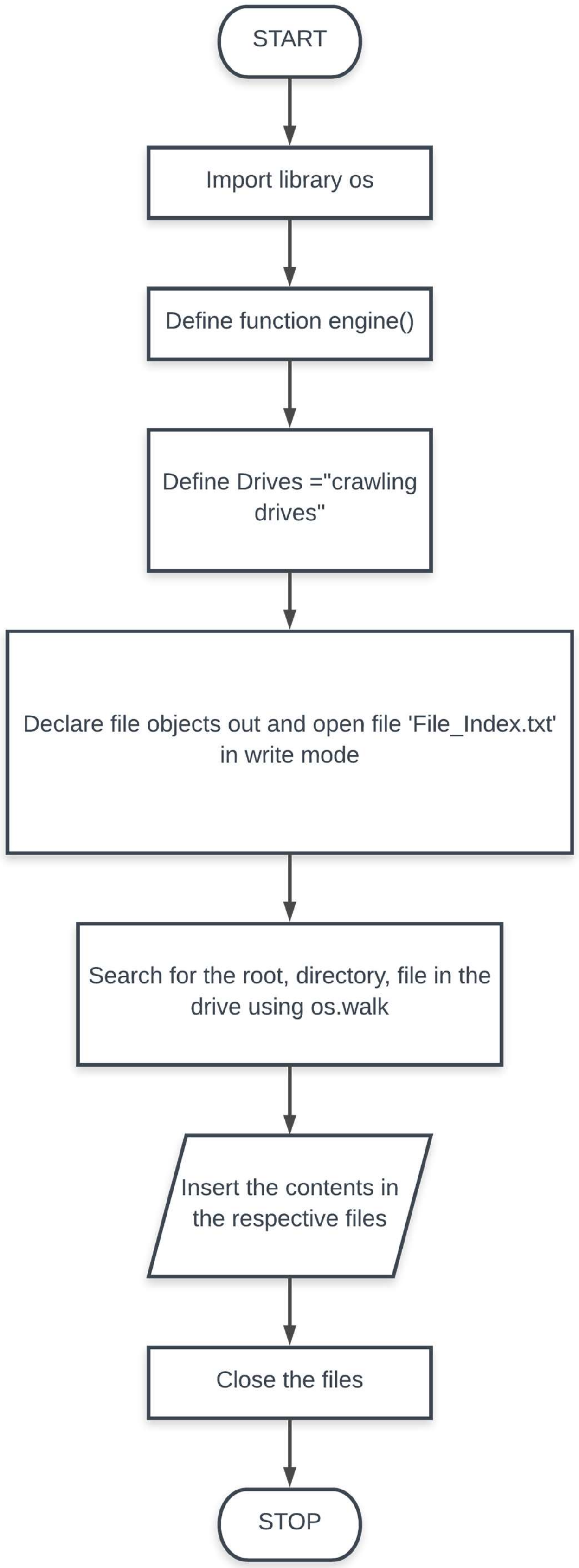


```
import os                       #To use the os operations.


def engine():        #Checks all the directories and files in a particular drive.
    drives = ["D:", "E:"]
    out = open('File_Index.txt', 'w+')       #Object for file.
    for i in drives:
        for root, dirs, files in os.walk(i):
            for name in files:
                out.write(name + "\n")
                out.write(os.path.join(root, name) + "\n")
            for name in dirs:
                out.write(name + "\n")
                out.write(os.path.join(root, name) + "\n")
    out.close()
```

In this module we have imported python library OS.

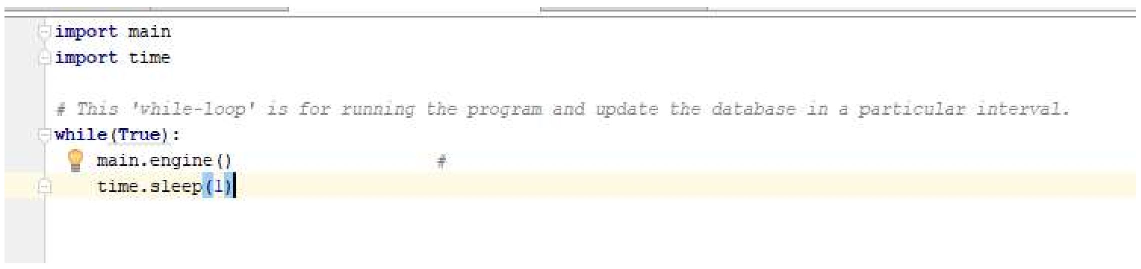
In this snippet, we have used function engine, that will be crawling the specified drives and storing the file locations in a text file.
Variables used:
Drives: it will crawl the drives specified in its scope
Out: it is a function object that will be used to open and write the file address in "File_Index.txt" text file
Root: it is used to store value of parent directory.
Dirs: it will be storing the information of directory
Files: it will be storing the location and address of the file in the sub-directory
os.Walk – it is OS library function that is doing the crawling part.


START
Import library os
Define function engine()
Define Drives ="crawling drives"
Declare file objects out and open file 'File_Index.txt' in write mode
Search for the root, directory, file in the drive using os.walk
Insert the contents in the respective files
Close the files
STOP

File: it is an object that will be opening the "output.txt" text file in read mode
Line: it will search for the specified phrase in file
If found, it will print that line with location.

Module 2:

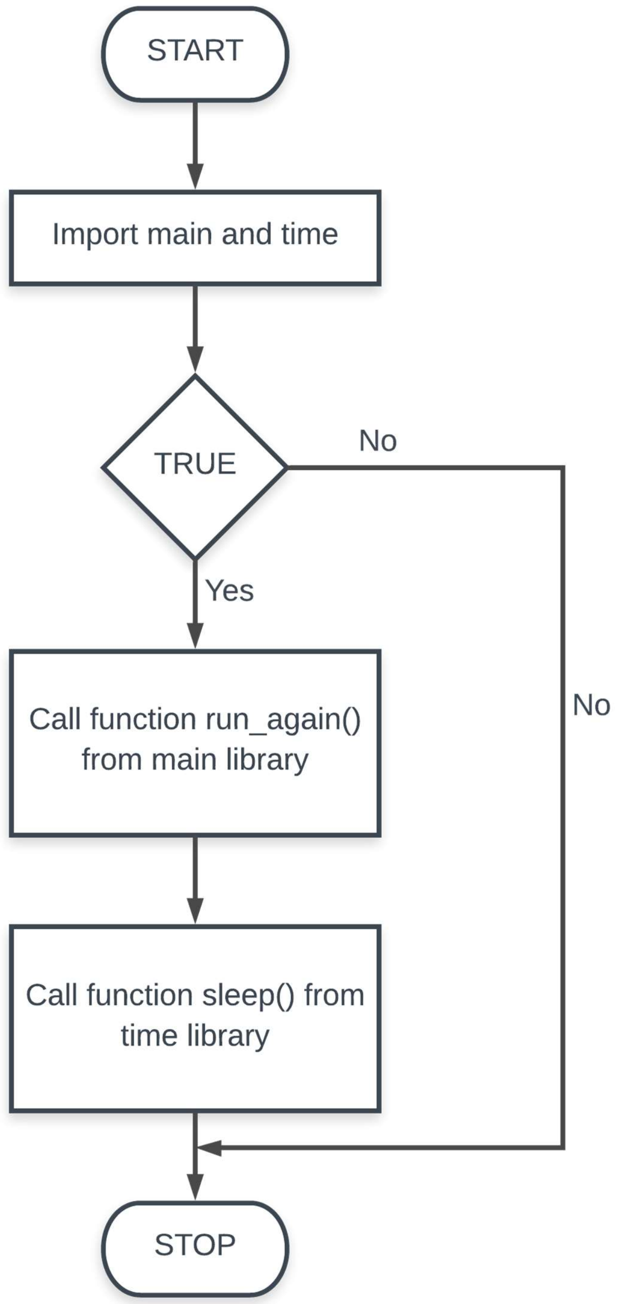
Recursive Execution



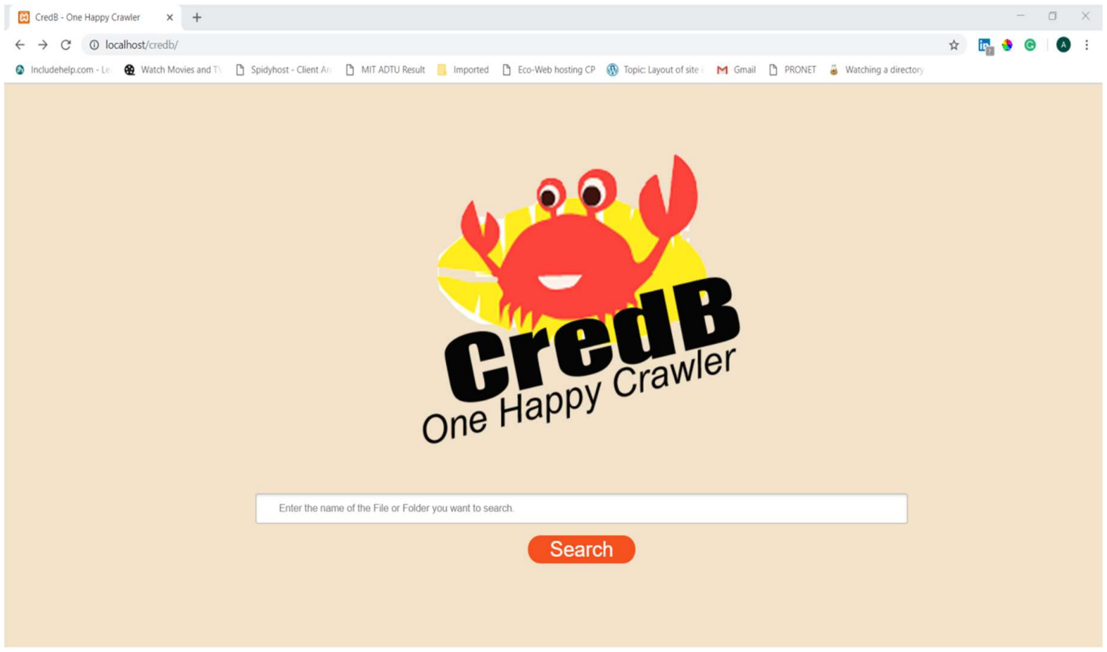
```
import main
import time

# This 'while-loop' is for running the program and update the database in a particular interval.
while(True):
    main.engine()                #
    time.sleep(1)
```

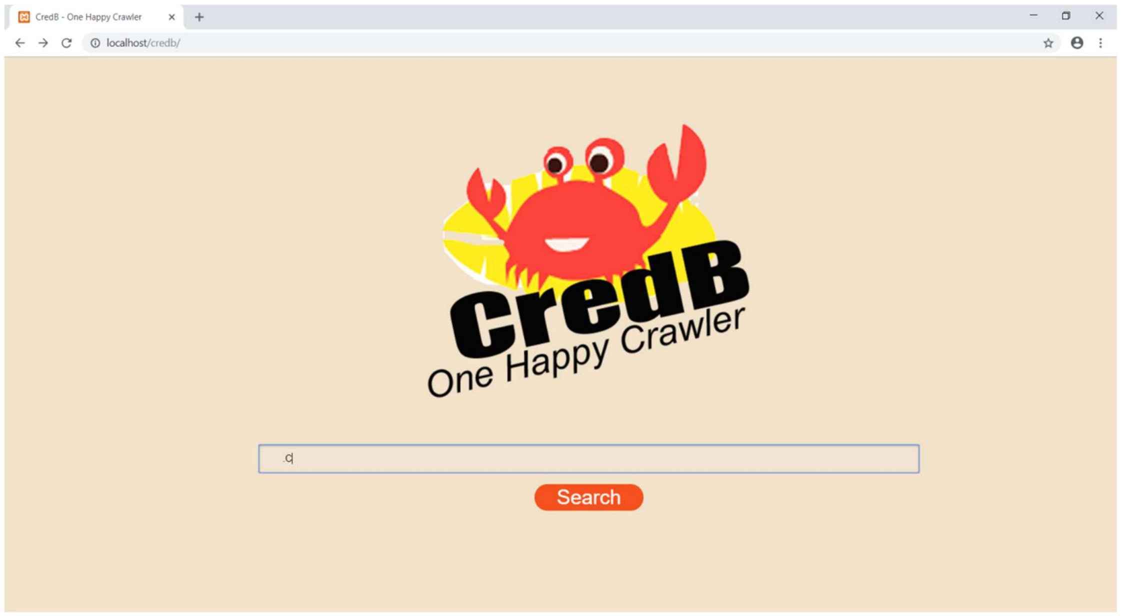
Time library is used in this module to use time function of system.
Here true Boolean operator will be used to check for the condition, since the condition will always remain true, hence the engine function defined in the main file will run after the specified interval in the function time.sleep().


START
Import main and time
TRUE
No
Yes
Call function run_again() from main library
No
Call function sleep() from time library
STOP

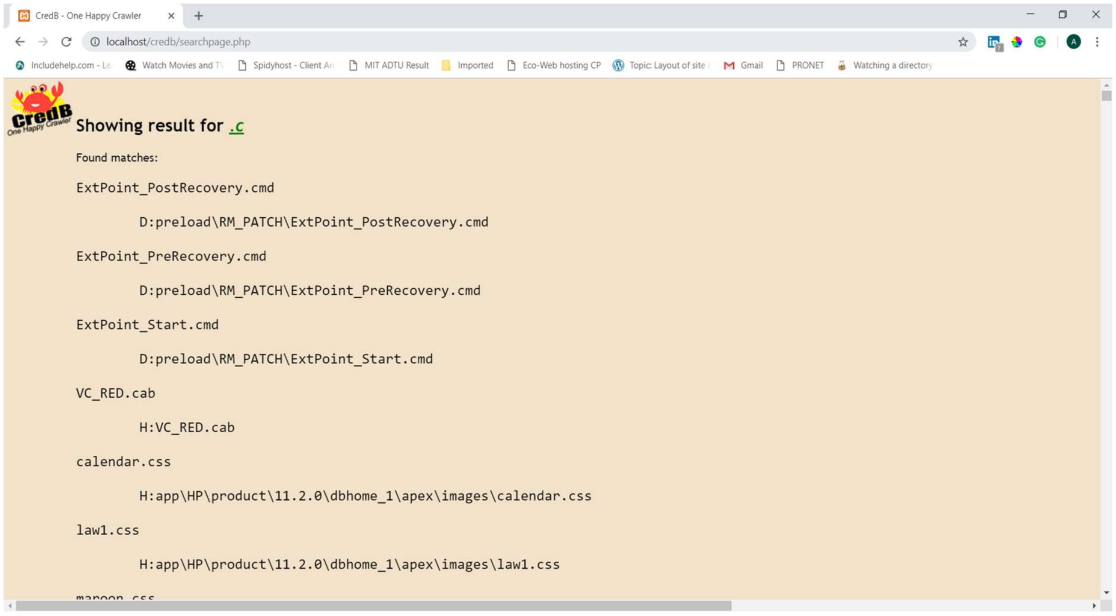
Output


CredB - One Happy Crawler
localhost/credb/
CredB
One Happy Crawler
Enter the name of the File or Folder you want to search.
Search

This is the front end of the directory crawler. Here the end-user will put the keyword to search



User entered ".c" for searching



Showing result for .c

Found matches:

```
ExtPoint_PostRecovery.cmd

        D:preload\RM_PATCH\ExtPoint_PostRecovery.cmd

ExtPoint_PreRecovery.cmd

        D:preload\RM_PATCH\ExtPoint_PreRecovery.cmd

ExtPoint_Start.cmd

        D:preload\RM_PATCH\ExtPoint_Start.cmd

VC_RED.cab

        H:VC_RED.cab

calendar.css

        H:app\HP\product\11.2.0\dbhome_1\apex\images\calendar.css

law1.css

        H:app\HP\product\11.2.0\dbhome_1\apex\images\law1.css
```

Here the end user has entered ".c" keyword and hence results matching the ".c" keyword in any of the statements in the file are displayed.

Source Code

Module 1: main.py

```
import os                         #To use the os operations.


def engine():       #Checks all the directories and files in a particular drive.
    drives = ["D:", "E:"]
    out = open('File_Index.txt', 'w+')       #Object for file.
    for i in drives:
        for root, dirs, files in os.walk(i):
            for name in files:
                out.write(name + "\n")
                out.write(os.path.join(root, name) + "\n")
            for name in dirs:
                out.write(name + "\n")
                out.write(os.path.join(root, name) + "\n")
    out.close()
```

Module 2: Recursive_execution.py

```
import main
import time

# This 'while-loop' is for running the program and update the database in a
particular interval.
while(True):
    main.engine()                   #
    time.sleep(1)
```

Front end code

Coded in PHP

Index.php

```
<html>
<head>
  <title>CredB - One Happy Crawler</title>
  <style>
.button {
  border-radius: 40px;
  background-color: #f4511e;
  border: none;
  color: #FFFFFF;
  text-align: center;
  font-size: 28px;
  padding: 2px;
  width: 150px;
  transition: all 0.5s;
  cursor: pointer;
  margin: 15px;
}

.button span {
  cursor: pointer;
  display: inline-block;
  position: relative;
  transition: 0.5s;

}

.button span:after {
  content: '\00bb';
  position: absolute;
  opacity: 0;
  top: 0;
  right: -20px;
  transition: 0.5s;
}

.button:hover span {
  padding-right: 25px;
}

.button:hover span:after {
  opacity: 1;
  right: 0;
}


.content{
  background-color: #F2E2CA;
  padding-left: 80px;
  margin: auto;
  max-width: 100%;
}

.aspect{
  padding-top: 50px;
}

#sbox{
  width:60%;
}
</style>
</head>
<link href="main_style.css" rel="stylesheet" type="text/css" />
<body>
  <div class=content>
    <div class="aspect">
    <h1>
      <center>
        <img src="credbp.png" alt="CredB-One Happy Crawler">
      </center>
    </h1>
    <form method="post" action="searchpage.php" name="registerform">
      <center>
      <input type=text id="sbox" name='query' onkeyup="showResult(this.value)"
autocomplete="off" placeholder="Enter the name of the File or Folder you want to search.">
      </input>
    </center>
      <center>
        <button class="button">
          <span>
            Search
          </span>
        </button>
      </center>
    </form>
  </div>
  </div>
</body>
</html>
```

Searchpage.php

```
<html>
<head>
 <title>CredB - One Happy Crawler</title>
<style>
        .se12{
                position: fixed;
                float: left;
        }
</style>
  </head>
<link href="main_style.css" rel="stylesheet" type="text/css" />

        <body>
        <img src = "credb-icon1.png" alt="CredB - One Happy Crawler" class="se12" />

                <div class=content style="background-color: #F2E2CA;
                                                                                padding-top:30px;
                                                                                padding-left:100px">
                        <?php

                                $q=htmlspecialchars($_POST['query']);
                                echo '<h2>Showing result for <i><u><font
color=green>'.$q.'</font></u></i></h2>';
                                $file = 'File_Index.txt';
                                $contents = file_get_contents($file);
                                $pattern = preg_quote($q, '/');
                                $pattern = "/^.*$pattern.*\$/m";
```

```
                    if(preg_match_all($pattern, $contents, $matches) ){
                      echo "Found matches:\n";
                      //echo "<pre>".implode("\n", $matches[0])."</pre>";
                      echo "<font size='5' face='Arial'>";
                      echo "<pre>".implode("\n\n" , $matches[0])."</pre>";
                    }
                    else{
                      echo "No matches found";
                    }
                ?>
            </div>
        </body>
</html>
```

Conclusion and Future Enhancements

This crawler will crawl all the system files present in any workstation that is attached to it, either through wired connection or wirelessly. This will not only reduce human efforts but will also save time.

Future enhancements of this module

For this project, we have kept out system drive out of crawling. In future extension of this application, we will be extending its reach to system drives too.

Later on, we will be extending this project to develop it into an offline search engine following a client server module where this crawler will serve as a full functioning search engine and will be used to fetch any data stored in any industrial sever that would employ it.

Glossary

Import – import module in python is similar to #include header file in c/c++. Python modules can get access to code from another module by importing the file/ function using from another module by importing the file function using import.

Time – library in python

OS - library in python

For...in.. – this is a conditional statement used for iterative purposes. Where for is used to take conditions, in statement is used to define the range.

While – while is a conditional statement used for iterative statements in python. While condition iterates a particular statement till a condition remains satisfied.

True - kind of similar to the true and false in real life, in python true and false statements like these acts as Boolean operator to check for a given condition.

Print – like printf statements in c language, print statement is used to print any specified piece of data.

References

www.stackoverflow.com

www.geeksforgeeks.org

www.google.com

www.pythonforbeginners.com

www.sanfoundry.com

www.jetbrains.com

www.ubuntuforums.org

www.tutorialspoint.com

www.udemy.com

www.coursera.org