**PROJECT ANALYSIS**

**Introduction**

The title of our project is **Wikipedia Search Automation**. It is prepared by **Naman Sharma**.

This project uses the Wikipedia API for Python. It also uses Flask, which is a web application framework in Python.

This project has 2 parts:-

* File Handling
* Web application

In File handling part, we provide a program that asks the user on the console to enter a keyword to be searched and fetches the data related to that keyword as it would be seen on the website of [www.wikipedia.org](http://www.wikipedia.org) .The data is returned from the module is then copied on to a text file and stored in the user’s system in a folder with the same name as the keyword searched.

In web application part, we have created a web page that lets the user input a keyword in a text box and then retrieves the data related to that keyword(if any) and then prints it on to the following web page in sections.

Existing System

Existing system can be found for the web application part at [www.wikipedia.org](http://www.wikipedia.org) . It served as an object of reference for this project.

**Hardware & software utilized**

**Hardware:** No special requirements are needed for the hardware aspect.

**Software:** Python 3.7, Notepad, IDLE(or any other IDE for Python), Google Chrome(or any other web browser).

**LANGUAGES USED**

Languages that were used in making this project were **Python**, **HTML**, **CSS** and **JavaScript**.

**Python modules used**

Python modules that were used in this project are: **Wikipedia**, **OS**, **Sys** and **Flask**.

Wikipedia module contains the builtin functions for the Wikipedia API.

OS module in Python provides functions for interacting with the operating system. This played an important role in the file handling part.

The sys module provides functions and variables used to manipulate different parts of the Python runtime environment.

Flask is a lightweight [WSGI](https://wsgi.readthedocs.io/) web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications.

**CODE**

File handling part:

import wikipedia

import os

import sys

def page\_content(page):

pg=wikipedia.page(page)

ttl=pg.title

url=pg.url

cont=pg.content

return ttl,url,cont

def page\_summary(page):

summ=wikipedia.summary(page)

return summ

def page\_search(page):

ser=wikipedia.search(page)

return ser

def main():

np=True

print('-'\*50)

print(' '\*25+'Wikipedia Search platform')

print(' '\*5+'-Prepared By the team of Nishtha Sharma,Manasvi Kumar and Naman Sharma')

while np==True:

page=input('\n\nEnter the keyword to be searched: ')

ser=page\_search(page)

os.mkdir(page)

os.chdir(page)

title,url,content=page\_content(page)

content=content.encode('unicode-escape').decode('utf-8')

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

f.write(content)

f.close()

url=url.encode('unicode-escape').decode('utf-8')

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

f.write(url)

f.close()

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

summ=page\_summary(page)

summ=summ.encode('unicode-escape').decode('utf-8')

f.write(summ)

f.close()

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

for line in ser:

f.write(line+'\n')

f.close()

os.chdir(os.pardir)

print('Folders Created!!')

res=input('Do you want to continue(y/n):')

if((res=='n')or(res=='N')):

np=False

elif((res=='y')or(res=='Y')):

np=True

else:

print('Invalid Entry!!!Session Terminated')

sys.exit(1)

main()

Web application part:

-python part

from flask import Flask,render\_template,redirect,session,request,abort

import wikipedia

app=Flask(\_\_name\_\_)

@app.route('/',methods=['POST','GET'])

def home():

return render\_template('main.html')

@app.route('/search',methods=['POST','GET'])

def search():

try:

if request.method=='POST':

key=str(request.form['nm'])

pg=wikipedia.page(key)

content=pg.content

content=content.encode('unicode-escape').decode('utf-8')

title=pg.title

title=title.encode('unicode-escape').decode('utf-8')

summ=wikipedia.summary(key)

summ=summ.encode('unicode-escape').decode('utf-8')

return render\_template('index.html',content=content,title=title,summ=summ)

else:

return render\_template('main.html')

except:

return "Page related to the entered Keyword not found"

if \_\_name\_\_=='\_\_main\_\_':

app.run(debug=True)

-Html part

(Main.html)

<html>

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<style>

form{

margin-left:35em

}

.center{

display:block;

margin-left:auto;

margin-right:auto;

width: 25%;

}

</style>

</head>

<body>

{% block body %}{% endblock %}

<img src="https://cdn2.downdetector.com/static/uploads/logo/wikipedia-logo\_1.png" class="center" alt="logo">

<h1 style="color:blue;" align="center">Wikipedia Search Automation</h1>

<h3 style="color:green;" align="center">-As prepared by the team of Nishtha Sharma, Manasvi Kumar and Naman Sharma</h3>

<form action="http://localhost:5000/search" method="post">

<p>Enter the keyword to be searched:</p>

<p><input type="text" name="nm" value={{request.form.key}}></p>

<p style="margin-left:4em"><input type="submit" value="search" /></p>

</form>

</body>

</html>

(Index.html)

{% extends 'main.html' %}

{% block body %}

<div class="block1">

<h1>Title: {{title}}</h1>

<h1>Summary:</h1>

<p><br>{{summ}}<br></p>

<h1>Contents:</h1>

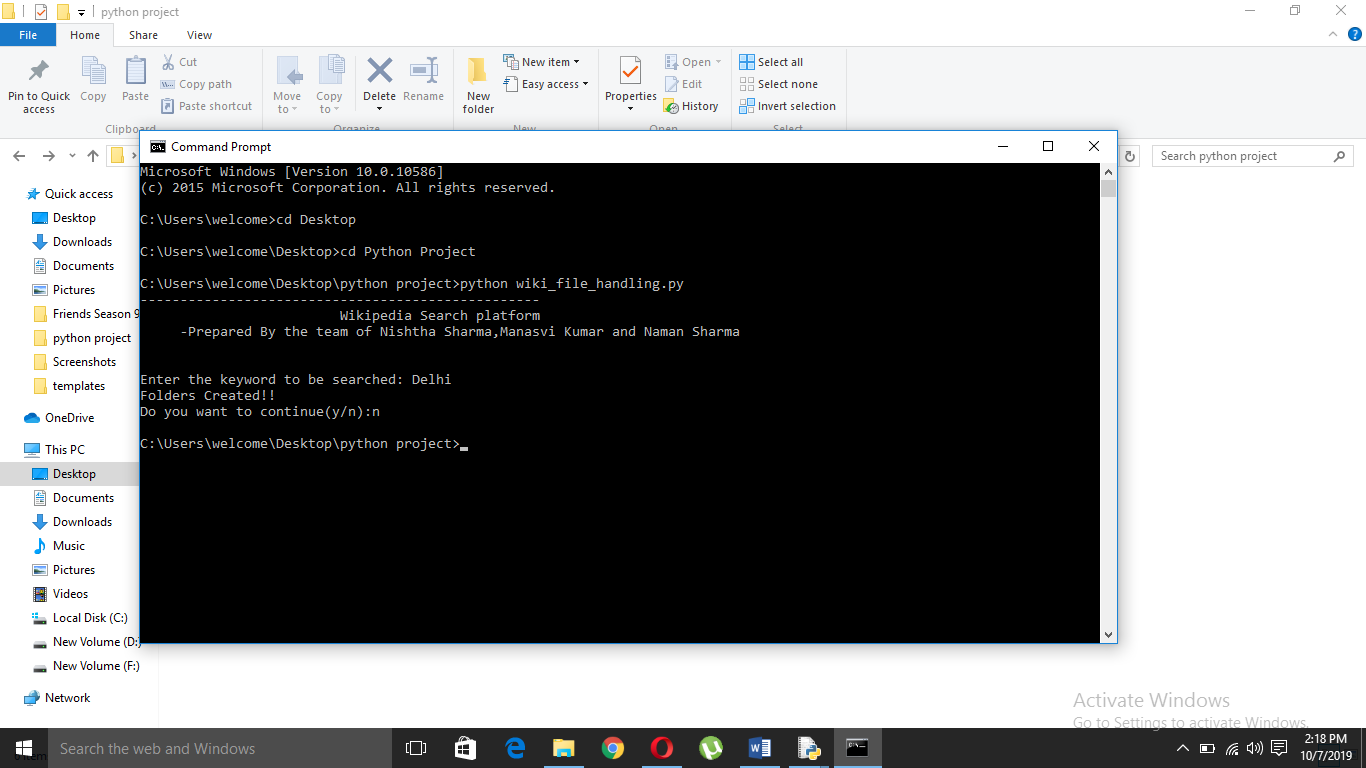
<p><br>{{content}}<br></p>

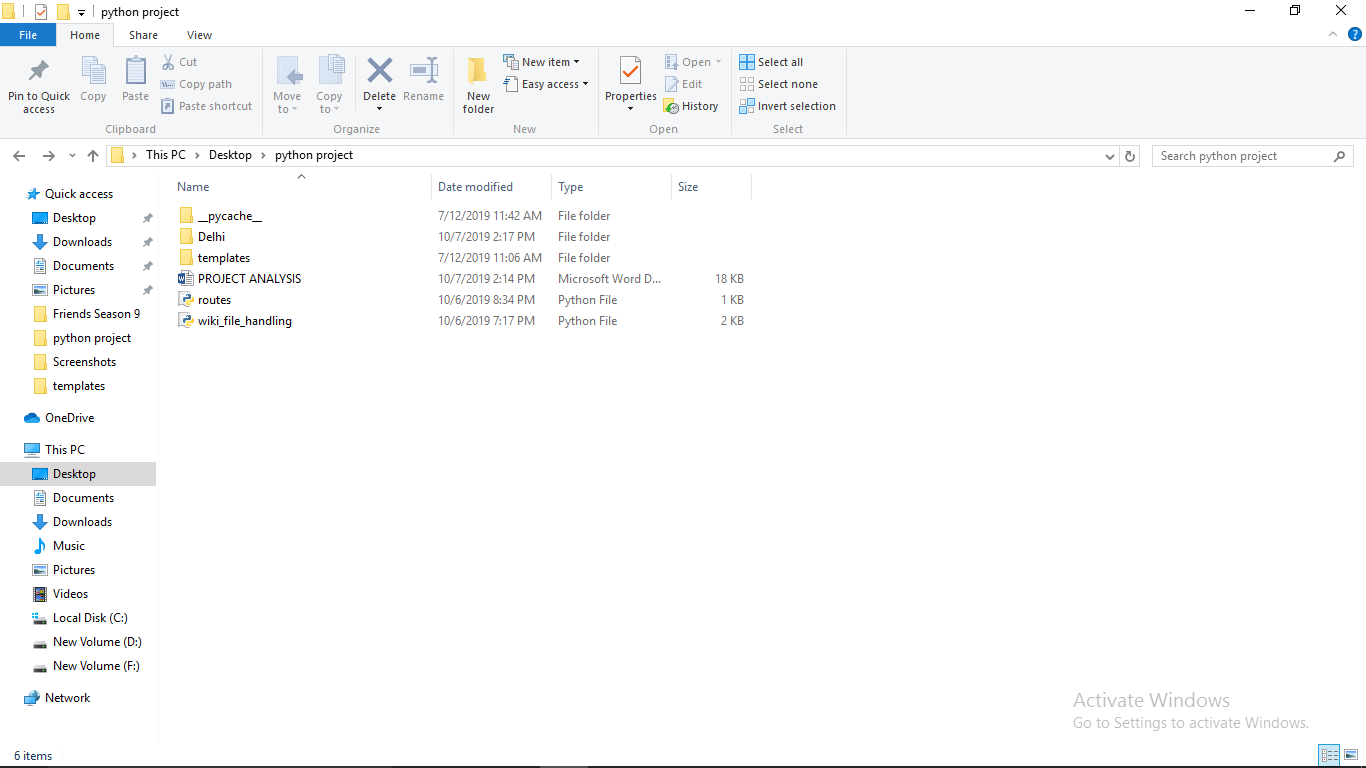
</div>

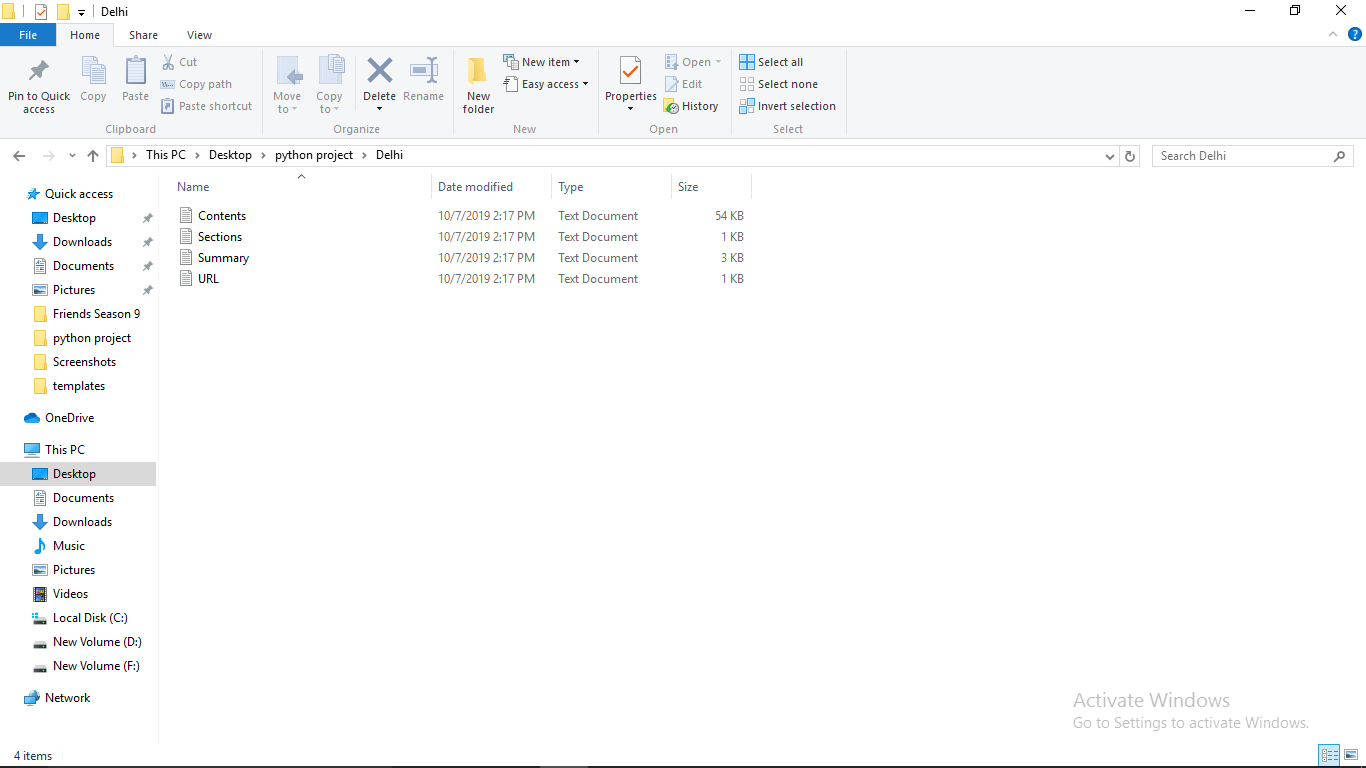
{% endblock %}

**Screenshots**

**File handling part:**







**Web application part:**

