Project 2 - Task Manager with User Authentication

By Abel Uche Ekwonyeaso

**Introduction:**

This project is about building a Task Management System with User Authentication using Python. This software will be useful for tracking users’ important tasks. The software will save all these details securely so that only an individual using it at a particular period can access his or her credentials when he or she inputs his or her account information. A project management program is one that is used to create, track, and manage projects tasks effectively. This task management system helps users achieve success. Python, with its simplicity and versatility combined with Tkinter's powerful GUI functionality, provides an excellent platform for systems development.

Tools and technologies used in this project

The tools and technologies used are as follows:

1. VS Code or Pycharm Community is used as code editor.
2. Python 3.x programming language.

Description of the Steps to Build the Task Management Systemin Python

Step 1: Create a folder called "Task Management System". Then, open it in VS Code or any code editor that is suitable.

Step 2: Create a Python file which is named "task\_manager.py".

Step 3: Set up the sign-up function. The registration function retrieves the username with which the user created the account and requires the password for the account. The code below will make this clear:

def signup():

print("Please enter the username by which you \

wanna access your account")

username = input("please enter here: ")

password = input("Enter a password: ")

Step 4: Create the "user\_info" function, which will receive the information from the "sign\_up" function and create a text that will save the user's information. The code below shows how the project works.

# pssd means password, ussnm is username

def user\_information(ussnm, pssd):

name = input("enter your name please: ")

address = input("your address")

age = input("Your age please")

ussnm\_ = ussnm+" task.txt"

f = open(ussnm\_, 'a')

f.write(pssd)

f.write("\nName: ")

f.write(name)

f.write('\n')

f.write("Address :")

f.write(address)

f.write('\n')

f.write("Age :")

f.write(age)

f.write('\n')

f.close()

def signup():

print("Please enter the username by which you\

wanna access your account")

username = input("please enter here: ")

password = input("Enter a password: ")

user\_information(username, password)

Step 5: Once the text file is created by the user information function, the next step is to code log-in function. The log-in function will take the username and ask for the password connected to it. Once the user enters the password the function will check if the password saved in the text file is the same as that entered. The code gives more explanation as follows:

def login():

print("Please enter your username ")

user\_nm = input("Enter here: ")

# Password as entered while logging in

pssd\_wr = (input("enterr the password"))+'\n'

try:

usernm = user\_nm+" task.txt"

f\_ = open(usernm, 'r')

# variable 'k' contains the password as saved

# in the file

k = f\_.readlines(0)[0]

f\_.close()

# Checking if the Password entered is same as

# the password saved while signing in

if pssd\_wr == k:

print(

"1--to view your data \n2--To add task \n3--Update\

task status \n4--View task status")

a = input()

else:

print("SIR YOUR PASSWORD OR USERNAME IS WRONG , Plz enter Again")

login()

except Exception as e:

print(e)

login()

Step 6: In this step, after the user signs in he or she is requested to log in to their account. This can be done by calling the log-in function at the end of the sign-in function. Hence, the sign-in function code is shown as follows:

def signup():

print("Please enter the username by which you wanna access your account")

username = input("please enter here: ")

password = input("Enter a password: ")

user\_information(username, password)

print("Sir please proceed towards log in")

login()

Step 7: In this step, the four important functions are completed as mentioned in the ‘login ‘block. Namely, function to view data, function to add a task to the data, function to update task status, and function to view task status. In this step also, the ‘login’ function is completed by completing the if-else part after taking input of the user’s demand, that is, the input to the variable a. Refer to the code below:

def login():

print("Please enter your username ")

user\_nm = input("Enter here: ")

# Password as entered while logging in

pssd\_wr = (input("enterr the password"))+'\n'

try:

usernm = user\_nm+" task.txt"

f\_ = open(usernm, 'r')

# variable 'k' contains the password as

# saved in the file

k = f\_.readlines(0)[0]

f\_.close()

# Checking if the Password entered is same

# as the password saved while signing in

if pssd\_wr == k:

print(

"1--to view your data \n2--To add task \n3--Update\

task \n4--VIEW TASK STATUS")

a = input()

if a == '1':

view\_data(usernm)

elif a == '2':

# add task

task\_information(usernm)

elif a == '3':

task\_update(user\_nm)

elif a == '4':

task\_update\_viewer(user\_nm)

else:

print("Wrong input ! ")

else:

print("SIR YOUR PASSWORD OR USERNAME IS WRONG")

login()

except Exception as e:

print(e)

login()

def view\_data(username):

pass

def task\_information(username):

pass

def task\_update(username):

pass

def task\_update\_viewer(username):

pass

As can be seen above, the pass command is used to enable writing of the function name and argument without the function body and also to prevent the error message from it in the code editor.

Step 8: Code the view data block as follows:

def view\_data(username):

ff = open(username, 'r')

print(ff.read())

ff.close()

Step 9: Code the number of tasks the user can add as follows:

def task\_information(username):

print("Sir enter n.o of task you want to ADD")

j = int(input())

f1 = open(username, 'a')

for i in range(1, j+1):

task = input("enter the task")

target = input("enter the target")

pp = "TASK "+str(i)+' :'

qq = "TARGET "+str(i)+" :"

f1.write(pp)

f1.write(task)

f1.write('\n')

f1.write(qq)

f1.write(target)

f1.write('\n')

print("Do u want to stop press space bar otherwise enter")

s = input()

if s == ' ':

break

f1.close()

The code above shows the flexibility the user has to add any number of task he or she wants to. This makes the program very much user-friendly.

Step 10: Updating the task status goes in the similar concept of text handling in python. Date time stamp for each task is used for tracking their status. That part of the text file will look like the following,

2024-12-01 14:44:02.851506

COMPLETED TASK

1,3,4

ONGOING TASK

2

NOT YET STARTED

5

def task\_update(username):

username = username+" TASK.txt"

print("Please enter the tasks which are completed ")

task\_completed = input()

print("Enter task which are still not started by you")

task\_not\_started = input()

print("Enter task which you are doing")

task\_ongoing = input()

fw = open(username, 'a')

DT = str(datetime.datetime.now())

fw.write(DT)

fw.write("\n")

fw.write("COMPLETED TASK \n")

fw.write(task\_completed)

fw.write("\n")

fw.write("ONGOING TASK \n")

fw.write(task\_ongoing)

fw.write("\n")

fw.write("NOT YET STARTED\n")

fw.write(task\_not\_started)

fw.write("\n")

Step 11: Code the task update viewer function. This function is as simple as the ‘view\_data’ function.

def task\_update\_viewer(username):

ussnm = username+" TASK.txt"

o = open(ussnm, 'r')

print(o.read())

o.close()

This puts to the end of the program. But before the end the most important task which is still left is coding the main function and controlling the flow of command of the program from the main function itself. This achieved in step 12.

Step 12: Coding the main function.

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

print("WELCOME TO ANURAG`S TASK MANAGER")

print("sir are you new to this software")

a = int(input("Type 1 if new otherwise press 0 ::"))

if a == 1:

signup()

elif a == 0:

login()

else:

print("You have provided wrong input !")

This marks the end of the code.