## LUCKNOW PUBLIC SCHOOL OF PROFESIONAL STUDIES

***"Excellence For All, Excellence From All"***



ACADEMIC YEAR **: 2020-21**

PROJECT REPORT ON

**“PyDiary : Digital Diary (GUI)”**

**ROLL NO : 49th**

**NAME : ALOK RANJAN**

**CLASS : XII - J**

**SUBJECT : COMPUTER SCIENCE**

**SUB CODE :**

# 

# CERTIFICATE

*THIS IS CERTIFIED TO BE THE BONAFIED WORK OF* ***ALOK RANJAN*** *IN THE COMPUTER LABORATORY DURING THE ACADEMIC SESSION 2018-19.*

*HE HAS SUCCESSFULLY COMPLETED HIS PROJECT IN THE COMPUTER PRACTICALS FOR THE AISSCE AS PRESCRIBED BY CBSE IN THE YEAR 2019-20.*

**.............…………………………**

INTERNAL EXAMINER **……………………………………**

EXTERNAL EXAMINER

|  |  |  |
| --- | --- | --- |
| **TABLE OF CONTENTS [ T O C ]** | | |
| **SNo.** | **DESCRIPTION** | **PAGE NO** |
| 01 | ACKNOWLEDGEMENT | **03** |
| 02 | INTRODUCTION | **06** |
| 05 | SOURCE CODE | **27** |
| 06 | OUTPUT (SCREENSHOTS) | **55** |
| 07 | INSTALLATION PROCEDURE | **65** |
| 08 | HARDWARE AND SOFTWARE REQUIREMENTS | **69** |
| 09 | BIBLIOGRAPHY | **70** |

## ACKNOWLEDGEMENT

Hello, Greetings, Thankyou for reviewing my project .

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

I express deep sense of gratitude to almighty God for giving me strength for the successful completion of the project.

I gratefully acknowledge the contribution of the individuals who contributed in bringing this project up to this level, who continues to look after me despite my flaws,

My sincere thanks to **Mrs.Ritu** Ma'am, Master In-charge, A guide, Mentor, who critically reviewed my project and helped in solving each and every problem, occurred during implementation of the project

I am overwhelmed to express my thanks to **The Administrative Officer, LPCPS** for providing me an infrastructure and moral support while carrying out this project in the school.

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. I am grateful for their constant support and help.

Thanking You

-**Alok Ranjan**

**PROJECT ON PYTHON-SQL : PYDIARY**

**INTRODUCTION**

The  **PyDiary**  is a program which is as same as a normal notebook or a diary, its a digital Diary , for storing your important notes, secrets , totally secured. It allows the user to create a user account, edit , save , view all notes , measure metrics about their notes and is GUI supported so that user can easily edit and view their notes anytime they want .

**Features :**

1. Allow the user to creat or enter into account using username and password, profile based set.
2. Contains GUI based text editor for ease of note writing
3. Password is will be not be visible while typing on the terminal, reducing the rist of password theft.
4. Create , Edit , View, Delete ,write your ideas in your notes ,in just few clicks, totally secured.
5. Contains a mini word processor for calculating number of words written, how many times a word is written.
6. Takes track and create a log of time and date of user logins , last time of note writing , edits etc.
7. Asks for the SQL server password before starting the signup/signin , reducing chances of data theft.
8. All your notes , at one place.

**INSPIRATION :**

*Making a diary and notes is a very good practice and everybody atleast make one in his/her lifetime. But the main problem is to sometimes things go wrong and your hardwork and hours of writing go to waste if the diary gets,water spills on or gets spoiled , stoled, or anything else ,which is really painful. The same happened with me , so I thought of using the knowledge of Python and Sql to make my own digitalized organized diary.*

***Here are some benefits of making a diary:***

**1. *Keeps your thoughts organized.***

### **2. Set & achieve your goals.**

### **3. Record ideas on-the-go.**

### **4. Relieve stress.**

### **5. Allow yourself to self-reflect.**

### **7. Boost your memory.**

### **8. Inspire creativity.**

# SOURCE CODE

**Creating a Python-SQL based project for making ORGANIZED DIARY NOTES MANAGEMENT.**

**Note :**

* Ask for SQL server password and initiate profile signup/signin process, if wrong close.
* Take note of login time and date and save in the log db for later use.
* Create user table if new user , or verify for the inputed password if already a user.
* Fetch all the profile details ie. Profile name, last login and all the available notes, show editor gui if user choose to create , edit note.

**SOURCE CODE:**

1. **import** pymysql
2. **import** datetime
3. **from** tkinter **import** \*
4. **import** tkinter as tk
5. **from** tkinter **import** scrolledtext
6. **from** tkinter **import** messagebox
7. **import** getpass

10. **def** getter():
11. getter.x = 1
13. **def** addlog(mycursor,usrid,conn):
14. mycursor.execute("USE " + usrid)
16. query = "UPDATE profile SET lastlog = %s ;"
17. value = (str(datetime.datetime.now()))
18. mycursor.execute(query, value)
19. conn.commit()
20. **print**(">Updated log")

23. **def** lastlogin(cursor,usrid):
24. cursor.execute("use "+usrid)
25. cursor.execute("select lastlog from profile ")
26. ll = cursor.fetchone()
27. **return** str(ll[0])
29. **def** fetchpass(cursor,usrid):
30. cursor.execute("use " + usrid)
31. cursor.execute("select password from profile ")
32. ll = cursor.fetchone()
33. **return** str(ll[0])


37. **def** connectDb():
38. **try**:
39. conn = pymysql.connect(
40. host='localhost',
41. user='user',
42. passwd='12345',
43. )
45. **print**(">DB Connected")
46. **return** conn
47. **except** Exception :
48. **print**(">Database not connected")

51. **def** checkUser(mycursor,usrid):
53. i = 0
54. l = []
55. **for** x **in** mycursor:
56. l.append(str(x).translate({ord(','): None, ord('('): None, ord(')'): None,
57. ord("'""'"): None})) # this removes the circular bracktes and comma from the sql output
59. if (l.\_\_contains\_\_(usrid)):
60. return True
61. else:
62. return False
64. def createTable(mycursor,notetitle):
65. sql\_command = """
66. CREATE TABLE %s (
67. password VARCHAR(20)
68. );
69. """ % notetitle
71. mycursor.execute(sql\_command)

74. def createUser(usrid, passwd , mycursor,conn):
75. if (usrid != "n"):
76. print("\n>Setting up your pyDiary profile.")
77. print(">Creating a db..")
78. query = "CREATE DATABASE "+usrid
79. mycursor.execute(query)
81. mycursor.execute("USE "+usrid )
83. sql\_command = "CREATE TABLE profile (username VARCHAR(30),password VARCHAR(30), lastlog VARCHAR(500) );"
85. mycursor.execute(sql\_command)

88. mycursor.execute("USE "+usrid)
90. query = """INSERT INTO profile (username,password,lastlog) VALUES (%s,%s,%s);"""
91. value = (usrid,passwd,str(datetime.datetime.now()))
92. mycursor.execute(query,value)
93. conn.commit()

96. print("> pyDiary ready!")
97. print("=============================( pyDiary Session : "+str(datetime.date.today())+" || UserID : "+usrid+" )=================>")
99. print(">Welcome to pyDiary : Your secrets, encrypted.")
100. HomePage(conn,mycursor,usrid)
101. #try:

104. # except Exception as e:
105. # print("Error initializing .... History feature will not work untill next restart.")
106. # print(e)



111. def NewNote(conn,title,mycursor,usrid):
113. ################################################### MAKE TABLE NOTE
114. mycursor.execute("USE " + usrid)
116. sql\_command = """CREATE TABLE %s ( log VARCHAR(500),Note VARCHAR(999999999));""" % title
118. mycursor.execute(sql\_command)
120. ################################################### MAKE TABLE NOTE
122. ################################### Window for writing
123. window = tk.Tk()
125. window.resizable(False,True)
127. window.title(title+" @"+usrid)
128. width =750
129. height = 500
130. window.geometry(str(width)+'x'+str(height))
132. txt = scrolledtext.ScrolledText(window, width=70, height=40)
133. txt.insert(INSERT,"Here you go...")
135. tk.Label(window,
136. text="pyDiary.",
137. font=("Times New Roman", 15),
138. background='green',
139. foreground="white").pack(padx=0, pady=5,side =tk.TOP)
140. tk.Label(window,
141. text=title,
142. font=("Times New Roman", 12),
143. background='white',
144. foreground="green").pack(padx=0, pady=5, side=tk.TOP)
146. def commit(text,comm,cursor,usrid):
148. try:
149. mycursor.execute("USE " + usrid)
151. query = "INSERT INTO " + str(title) + " (log,Note) VALUES (%s,%s);"
152. value = (str(datetime.datetime.now()), text)
153. mycursor.execute(query, value)
154. conn.commit()
155. if messagebox.showinfo("Notes Saved", "Your notes were successfully saved!"):
156. print(">Notes Saved and Closed.")
157. window.destroy()
158. except Exception:
159. messagebox.showwarning("Notes Not Saved", "Your notes were not saved due to some error !")






167. btnSave = Button(window, text="Save & Exit",fg="white",bg="green",width = 20, height = 2,command = lambda :commit(str(txt.get('1.0',END)),conn,mycursor,usrid))
169. def word\_count(str):
170. counts = dict()
171. words = str.split()
173. for word in words:
174. if word in counts:
175. counts[word] += 1
176. else:
177. counts[word] = 1
179. return counts


183. def clicked(text):
184. words = len(text.split())
185. wc = word\_count(text)
186. messagebox.showinfo("Metrics","Word Count :\n\n"+str(wc)+"\n\nTotal Words :"+str(words))
188. btnClose = Button(window, text="Metrics",fg= "green",bg="WHITE", width=20, height=2,command = lambda :clicked(str(txt.get('1.0',END))))
190. # btnSave.grid(column=2, row=0)
191. # btnClose.grid(column=2, row=1)
192. txt.pack(padx=5, pady=10, side=tk.LEFT)
193. btnSave.pack(padx=5, pady=20, side=tk.BOTTOM)
194. btnClose.pack(padx=5, pady=5, side=tk.BOTTOM)
196. btnClose.pack()

199. window.mainloop()
200. print("\n========================<Session closed>=================>")
202. choose(conn,mycursor,usrid)



207. def OpenNote(conn , mycursor, usrid, note\_data, title):
208. window = tk.Tk()
210. window.resizable(False, True)
212. window.title(title + " @" + usrid)
213. width = 750
214. height = 500
215. window.geometry(str(width) + 'x' + str(height))
217. txt = scrolledtext.ScrolledText(window, width=70, height=40)
218. note\_data = str(note\_data).translate({ord(','): None, ord('('): None, ord(')'): None, ord("'"): None})
219. txt.insert(INSERT, str(note\_data).replace("\\n"," \n "))
221. tk.Label(window,
222. text="pyDiary.",
223. font=("Times New Roman", 15),
224. background='green',
225. foreground="white").pack(padx=0, pady=5, side=tk.TOP)
226. tk.Label(window,
227. text=title,
228. font=("Times New Roman", 12),
229. background='white',
230. foreground="green").pack(padx=0, pady=5, side=tk.TOP)

233. **def** commit(text, conn, mycursor, usrid):
235. **try**:
236. mycursor.execute("USE " + usrid)
238. query = "UPDATE " + title + " SET log = %s, Note = %s ;"
239. value = (str(datetime.datetime.now()), text)
240. mycursor.execute(query, value)
241. conn.commit()

244. **print**(">Note's Progress Saved.")


248. **except** Exception as e:
249. messagebox.showwarning("Notes Not Saved", "Your notes were not saved due to some error !\n "+str(e))
251. btnSave = Button(window, text="Save progress", fg="white", bg="green", width=20, height=2,
252. command=**lambda**: commit(str(txt.get('1.0', END)), conn, mycursor, usrid))

255. **def** clicked(text):
256. **try**:
257. mycursor.execute("USE "+usrid )
259. query = "UPDATE "+title+" SET log = %s, Note = %s ;"
260. value = (str(datetime.datetime.now()), text)
261. mycursor.execute(query, value)
262. conn.commit()
263. **if** messagebox.showinfo("Notes Saved", "Your note's progress have been saved!"):
264. **print**(">Note's Progress Saved .")
265. window.destroy()




271. **except** Exception as e:
272. messagebox.showwarning("Notes Not Saved", "Your notes were not saved due to some error !\n "+str(e))


276. btnClose = Button(window, text="Save & Exit", fg="green", bg="WHITE", width=20, height=2,
277. command=**lambda** :clicked(str(txt.get('1.0',END))))
279. # btnSave.grid(column=2, row=0)
280. # btnClose.grid(column=2, row=1)
281. txt.pack(padx=5, pady=10, side=tk.LEFT)
282. btnSave.pack(padx=5, pady=20, side=tk.BOTTOM)
283. btnClose.pack(padx=5, pady=5, side=tk.BOTTOM)
285. btnClose.pack()
287. window.mainloop()
288. **print**("\n========================<Session closed>=================>")
290. choose(conn,mycursor,usrid)
292. **def** choose(conn,mycursor,usrid):
293. mycursor.execute("use " + usrid)
294. **print**("\n>What to do today ?")
295. **print**(""
296. "\n1.NEW NOTE\n"
297. "2.OPEN NOTE\n"
298. "3.DELETE NOTE\n"
299. "4.LOG OUT\n"
300. )
301. inp = input(">Select : ")
303. **if** (int(inp) == 1):
304. title = input(">Note Title : ")
305. title = "`" + title + "`"
306. NewNote(conn, title, mycursor, usrid)
308. **if** (int(inp) == 2):
309. **print**("\n>Open Notes Selected.")
310. mycursor.execute("use " + usrid)
311. mycursor.execute("show tables;")
312. **print**("\n>Notes Library:\n")
313. n = 1
314. l = []
315. **for** i **in** mycursor:
316. tbname = str(i).translate({ord(','): None, ord('('): None, ord(')'): None, ord("'""'"): None})
317. if tbname != "profile":
318. print(n, "-> ", tbname)
319. if tbname.\_\_contains\_\_(" "):
320. tbname = '`' + tbname + '`'
321. l.append(tbname)
322. else:
323. l.append(tbname)
324. n = n + 1
325. if (len(l)==0):
326. print("Note Library Empty, add some notes.")
327. print("\n========================<Session closed>=================>")
328. choose(conn,mycursor,usrid)
329. else:
330. inp = input("\n\nSelect Note (By S.No): ")
331. note = l[int(inp) - 1]
333. mycursor.execute("select Note from " + note + ";")
334. note\_data = mycursor.fetchone()
335. OpenNote(conn, mycursor, usrid, note\_data, note)

338. if (int(inp)==3):
339. print("\n>Delete Notes Selected.")
340. mycursor.execute("use " + usrid)
341. mycursor.execute("show tables;")
342. print("\n>Notes Library:\n")
343. n = 1
344. l = []
345. for i in mycursor:
346. tbname = str(i).translate({ord(','): None, ord('('): None, ord(')'): None, ord("'"): None})
347. **if** tbname != "profile":
348. **print**(n, "-> ", tbname)
349. **if** tbname.\_\_contains\_\_(" "):
350. tbname = '`' + tbname + '`'
351. l.append(tbname)
352. **else**:
353. l.append(tbname)
354. n = n + 1
356. inp = input("\n\nSelect Note (By S.No): ")
357. note = l[int(inp) - 1]
358. **while**(True):
359. confirm = input("Write '" + note + "' to confirm : ")
361. **if** (confirm == note):
362. query = "DROP TABLE %s;" % note
363. mycursor.execute(query)
364. **print**("Deleted Note :"+note)
365. **print**("\n========================<Session closed>=================>")
366. choose(conn,mycursor,usrid)
367. **if** int(inp) == 4:
368. **print**(">Logging out.")
369. auth(mycursor,conn)


373. **def** HomePage(conn,mycursor,usrid):
374. **print**("\n========================<Profile>==============")
375. **print**("HELLO,"+str(usrid).upper()+"!")
376. ll = lastlogin(mycursor,usrid)
377. **print**("Last Login : ",ll)
378. **print**("=========================<Profile>===============\n")
379. choose(conn,mycursor,usrid)



384. **def** auth(mycusor,conn):
385. **print**("\n========================<pyDiary Auth>======================")
386. **while**(True):
387. usrId = input("\nEnter UserID or type 'n' for new user :")
389. **if** usrId != 'n':
390. mycusor.execute("show databases;")
391. **if** (checkUser(mycusor,usrId)):
392. pass\_in = getpass.getpass(prompt="Enter your Password : ")
393. mycusor.execute("use "+usrId)
394. pass\_db =fetchpass(mycusor,usrId)
396. **if** (pass\_in == pass\_db):
397. **print**(">Login Successful , Welcome.")
398. **print**(">pyDiary ready!")
399. **print**("\n=============================( pyDiary v1.0a Session : " + str(
400. datetime.date.today()) + " || UserID : " + usrId + " )=================>")
402. **print**(">Welcome to pyDiary : Your secrets, encrypted.")
403. addlog(mycusor,usrId,conn)
405. HomePage(conn, mycusor, usrId)
406. **else**:
407. **print**("Profile Locked! , you can't proceed !")
408. **break**


412. **else**:
413. n = input(">UserID not found , Make new account(y/n) : ")
414. **if** (n=="y"):
415. **continue**
416. **else** :
417. **print**("OK, closing pyDiary.")
418. quit()
419. **break**


423. **else**:
424. userId = input("Enter your new UserID (Example: user123) :")
425. passwd = getpass.getpass(prompt="Enter your new Password : ")
426. userId = '`'+userId+'`'
427. **print**(usrId)
428. createUser(userId,passwd,mycusor,conn)







437. **def** Dinit():
439. **try**:
440. **print**(">Enter SQL server password (default is '' , just press enter ) :")
441. db = getpass.getpass()
442. conn = pymysql.connect("localhost", "root", str(db))
443. **print**(">DB Connected")
444. cursor = conn.cursor()
445. cursor.execute("show databases;")
446. auth(cursor, conn)
448. **except** Exception as e:
449. **print**("ERROR, DB NOT CONNECTED : RUN THE SERVER : "+str(e))

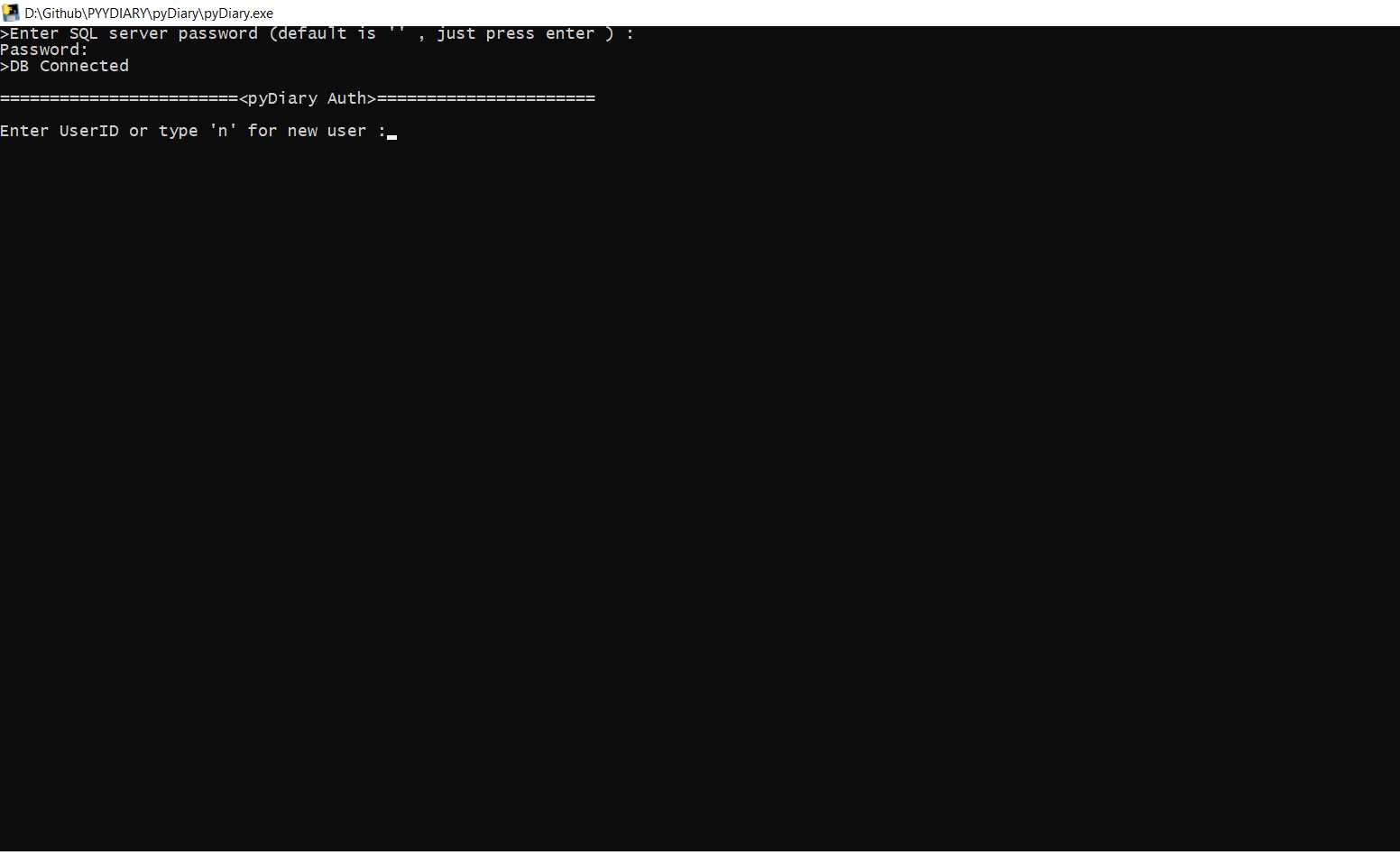




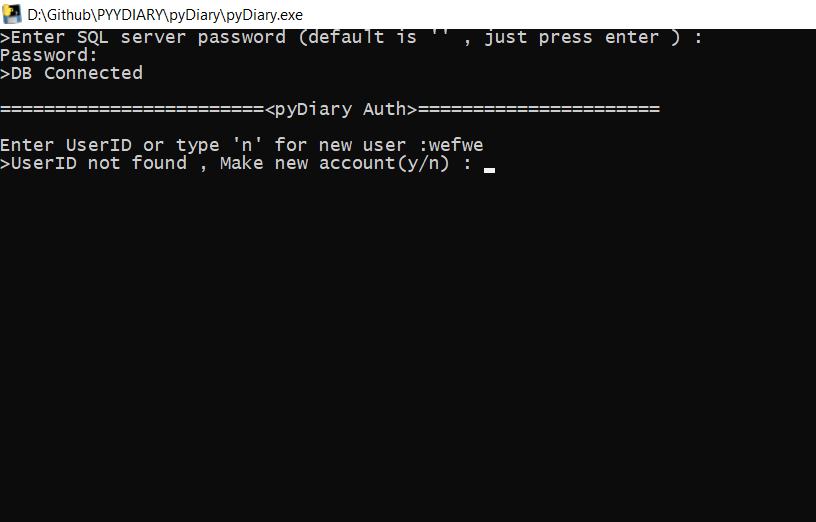


458. Dinit()
459. #HomePage("efs","aprit")
460. #OpenNote()

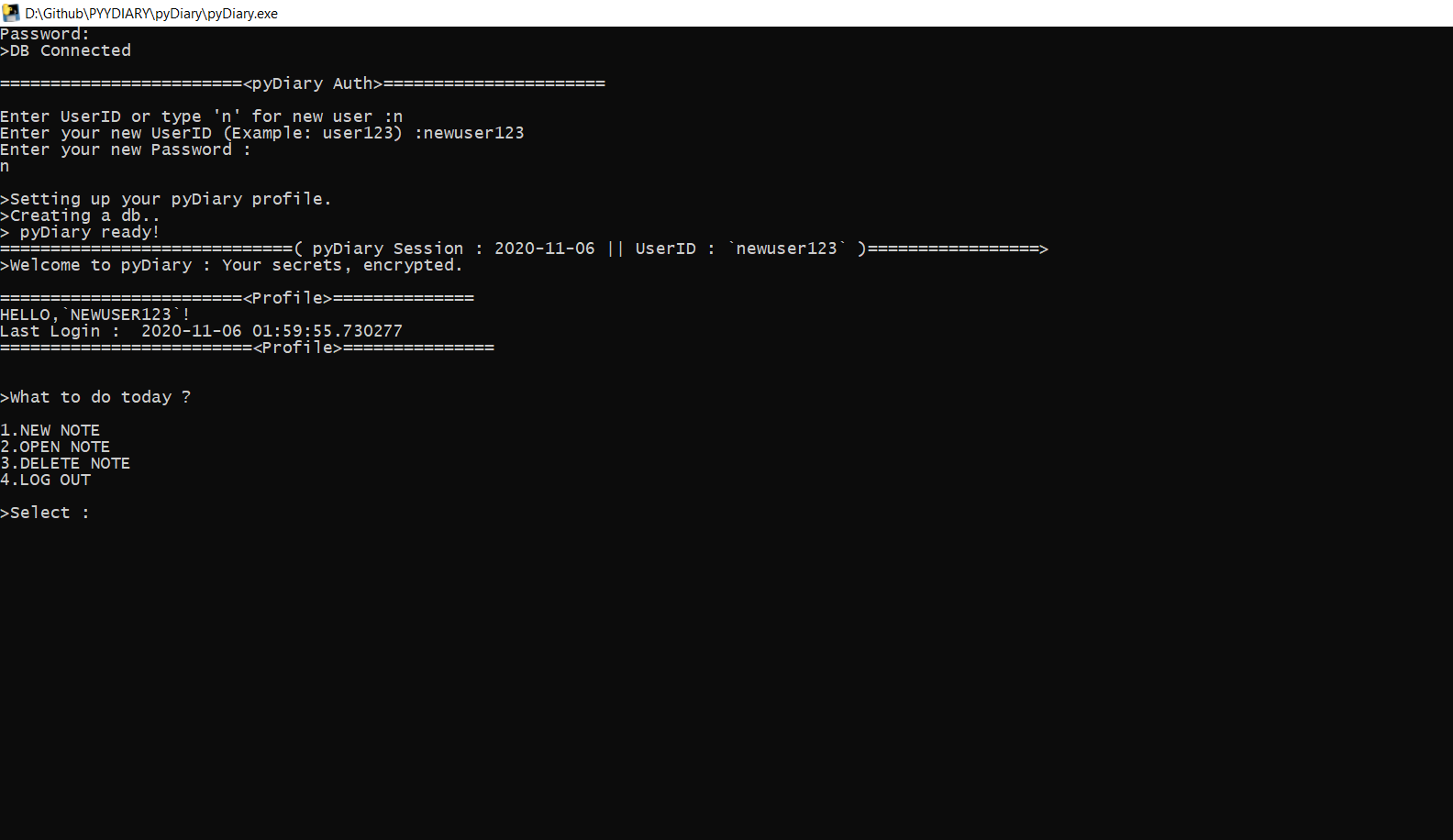
**WALKTHROUGH (SCREENSHOTS ) :**



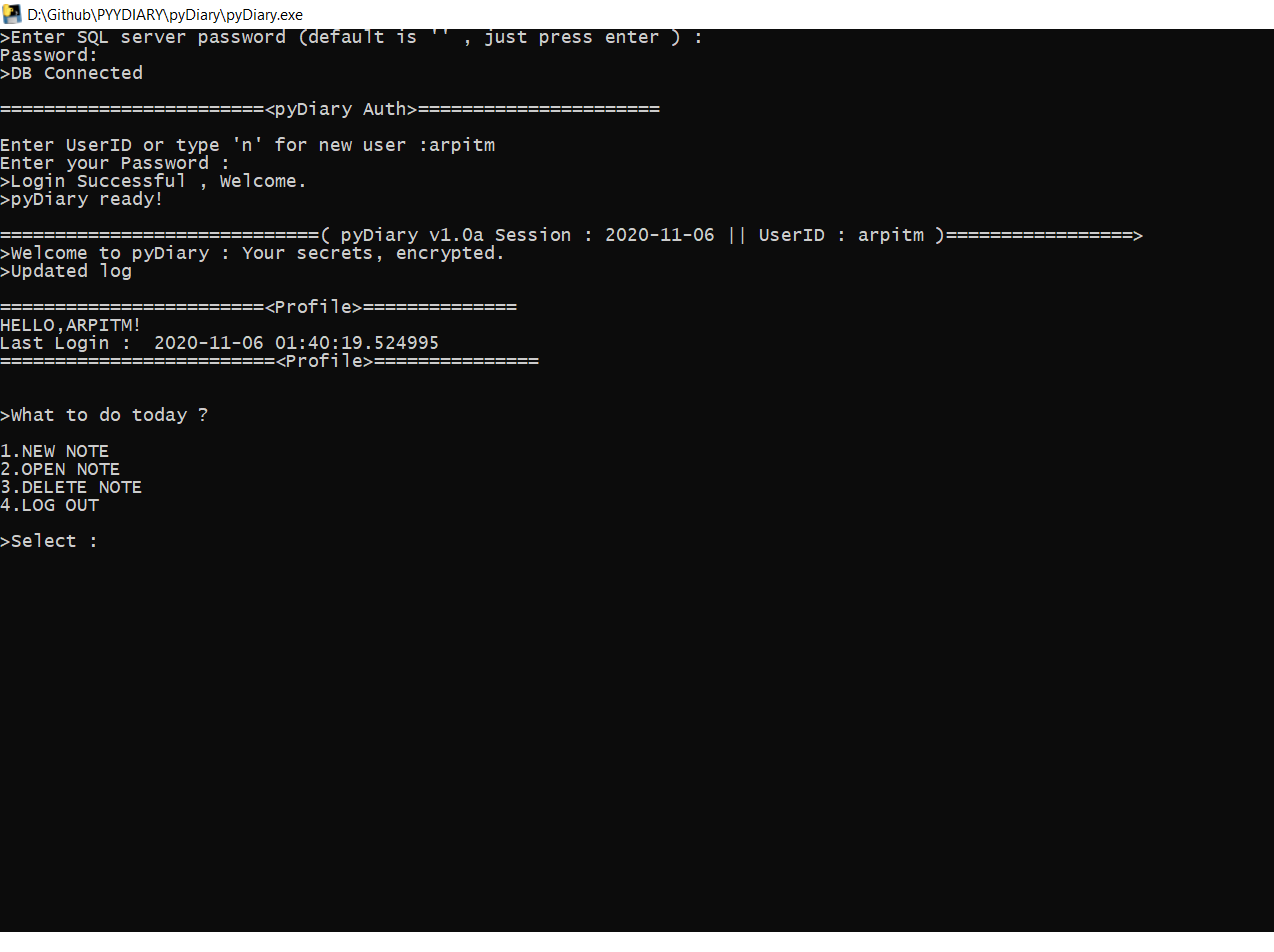
Taking SQL server's password, ensure that a sql server is running in the background.



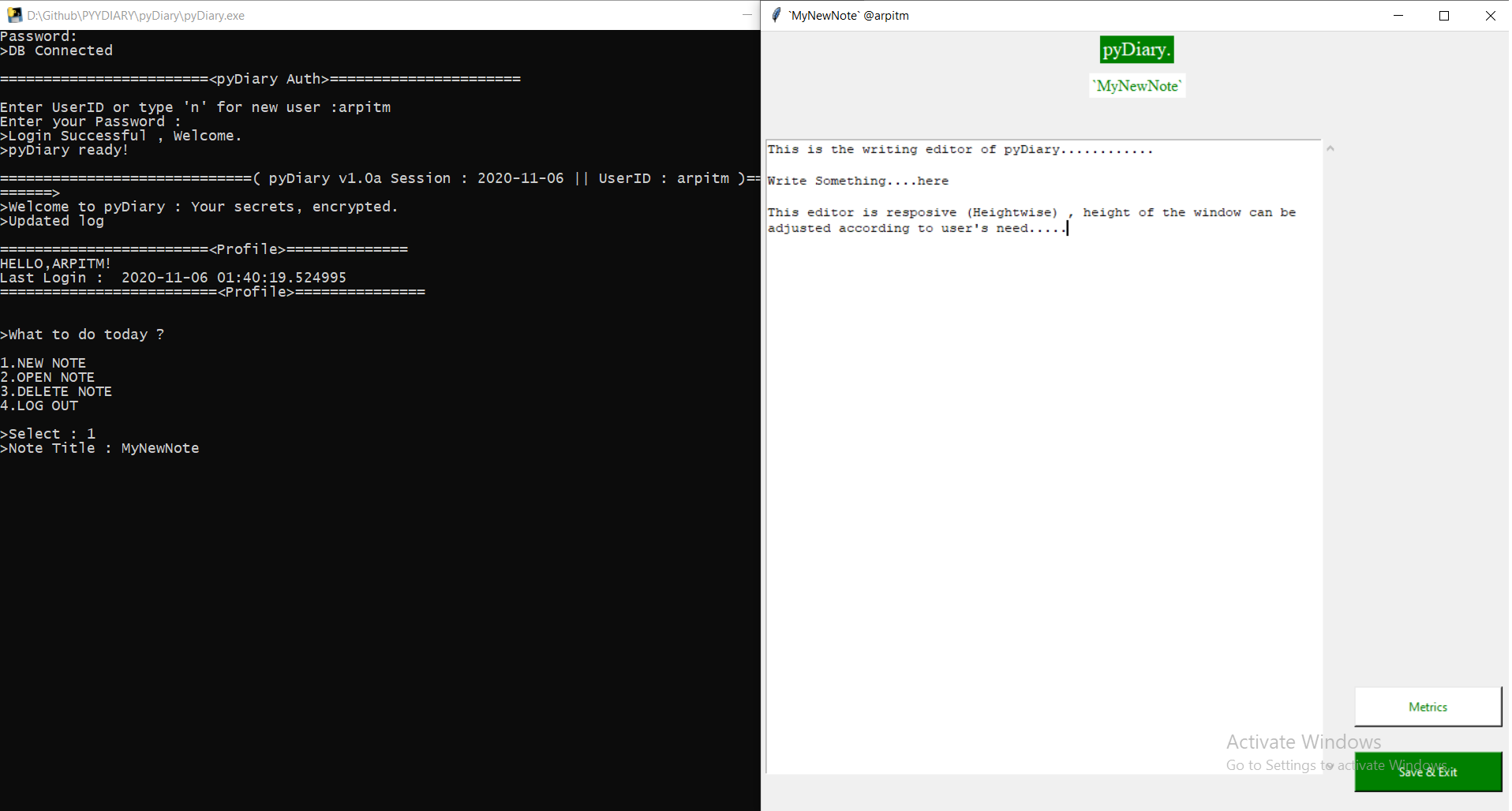
Failed Auth, no user was present with the username entered.



Created a new user named newuser123.

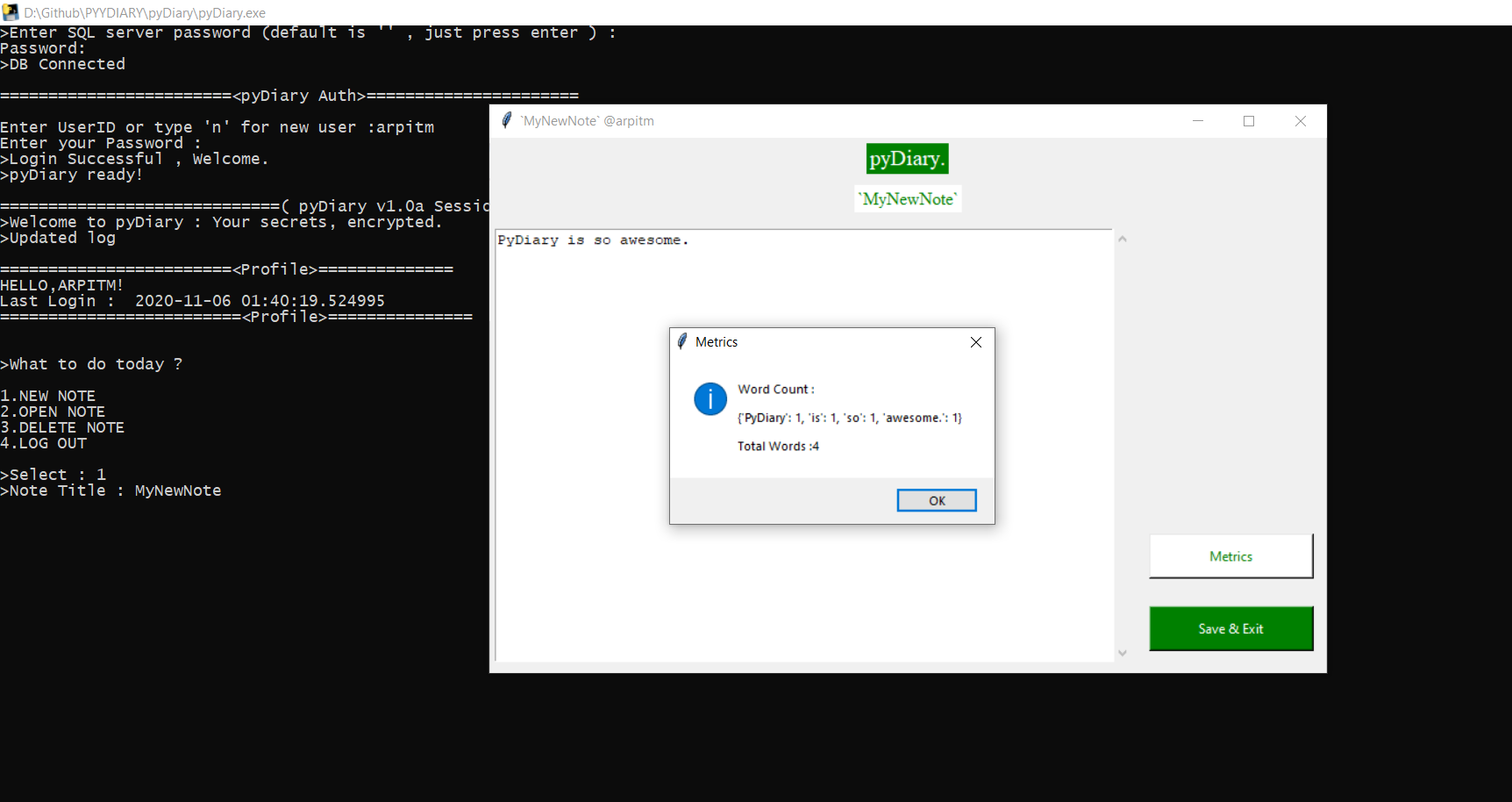


Logined as a existing user named 'arpitm', pyDiary collected last time login data and fetch all notes on this profile.

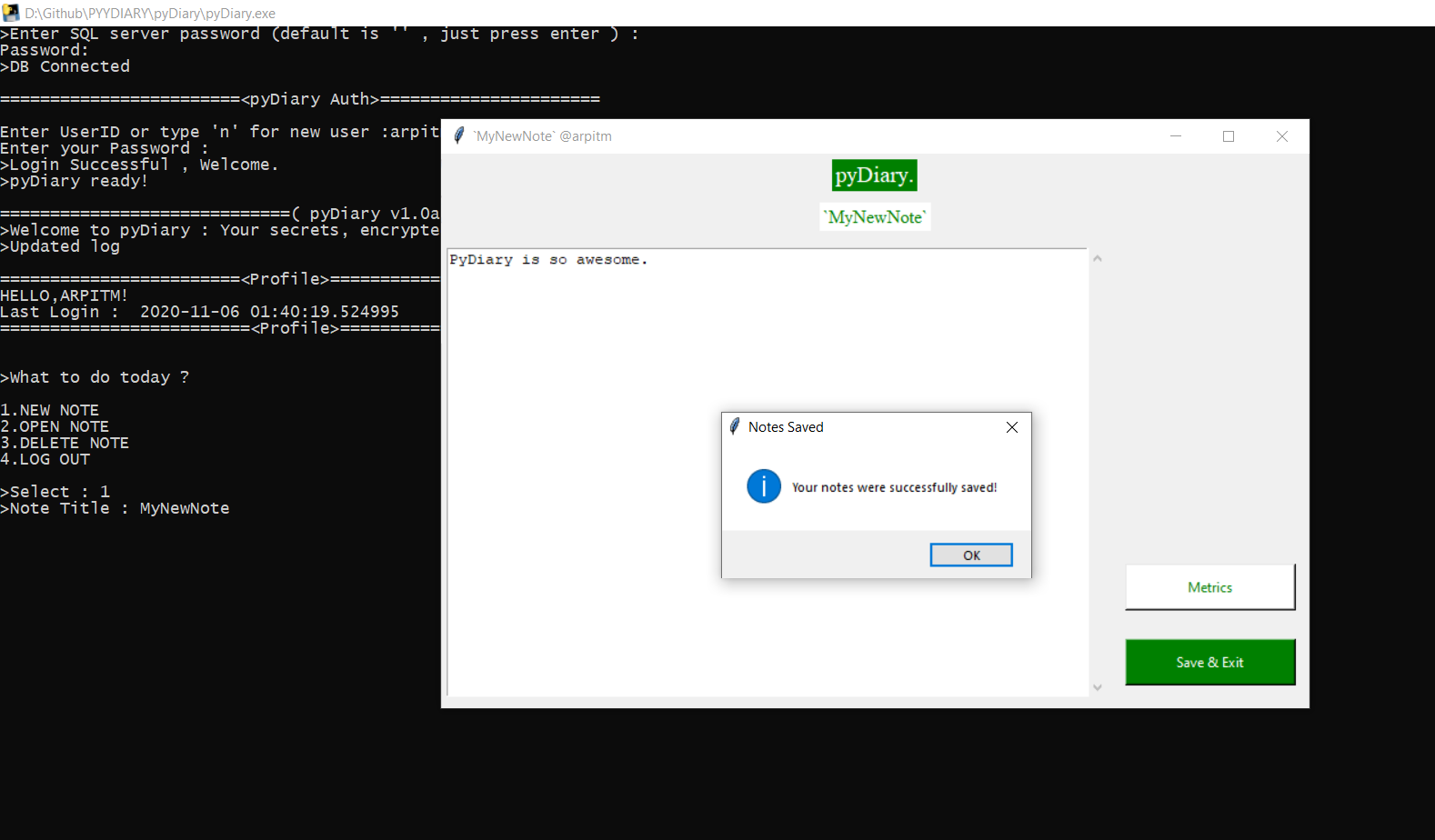


Chosed 1 , To create a new note , named : 'MyNewNote'

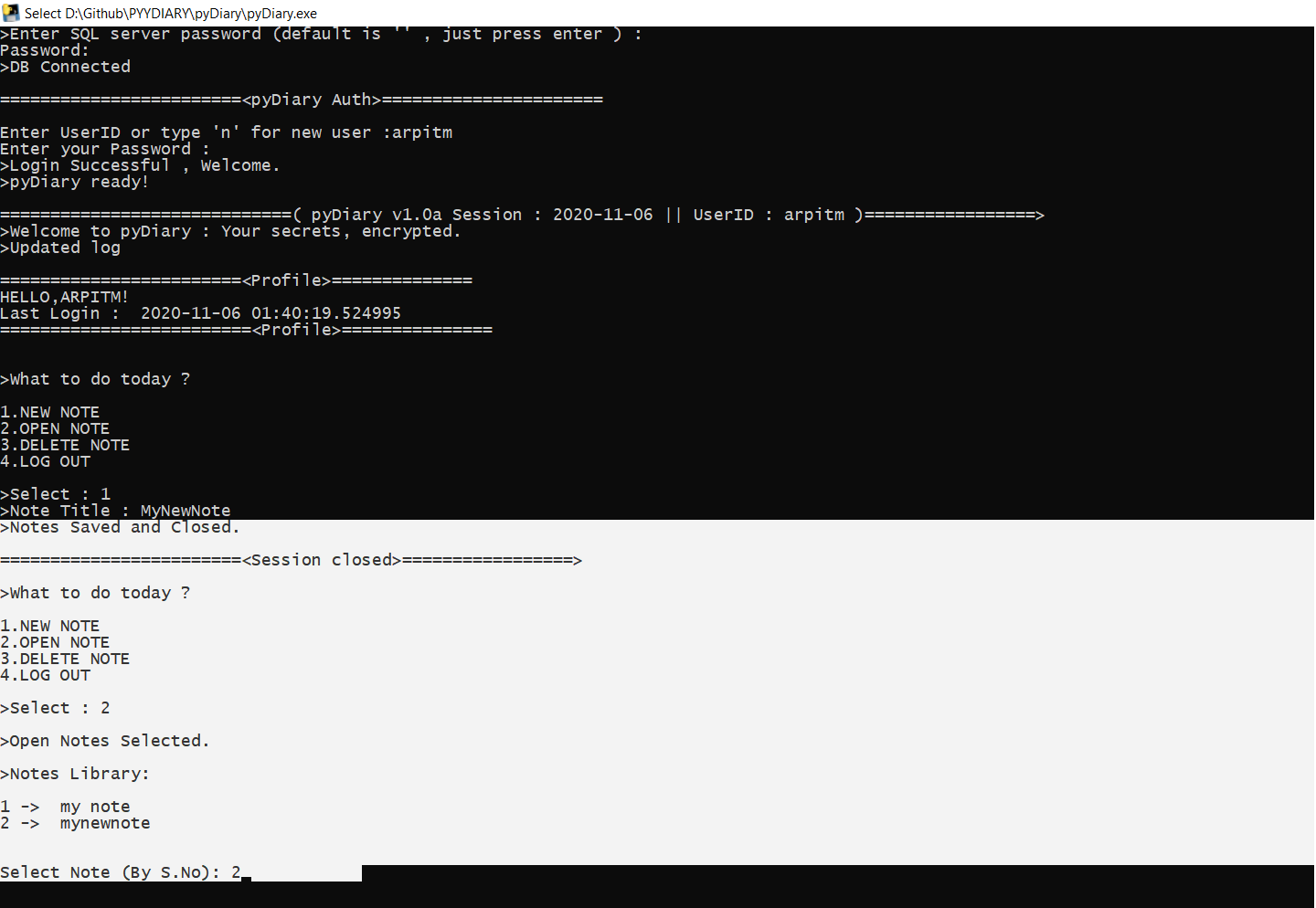
PyDiary gui editor opened showing writable space..



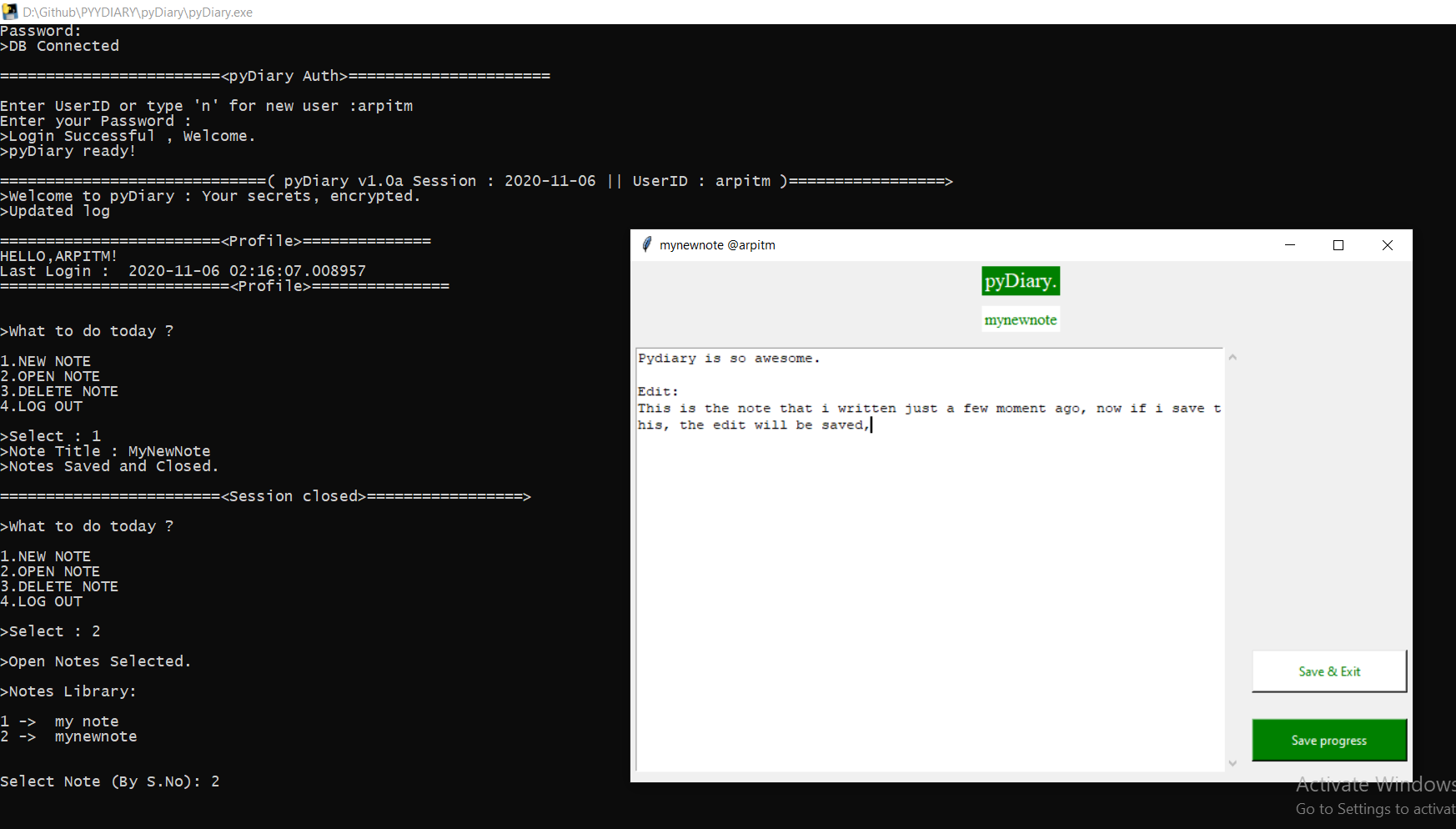
Mini Word processor of pyDiary.



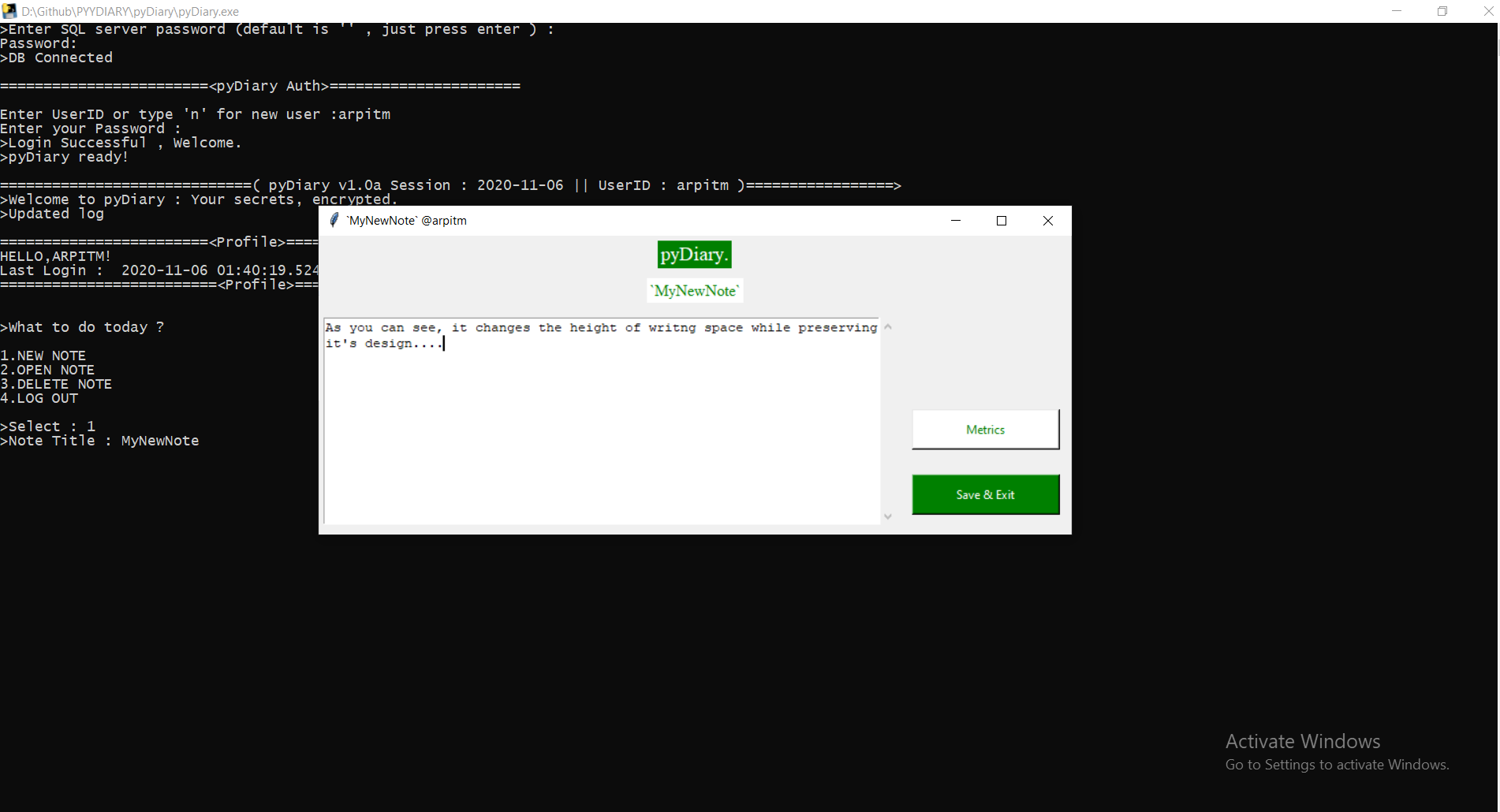
Notes are writen and commited in the SQL database.

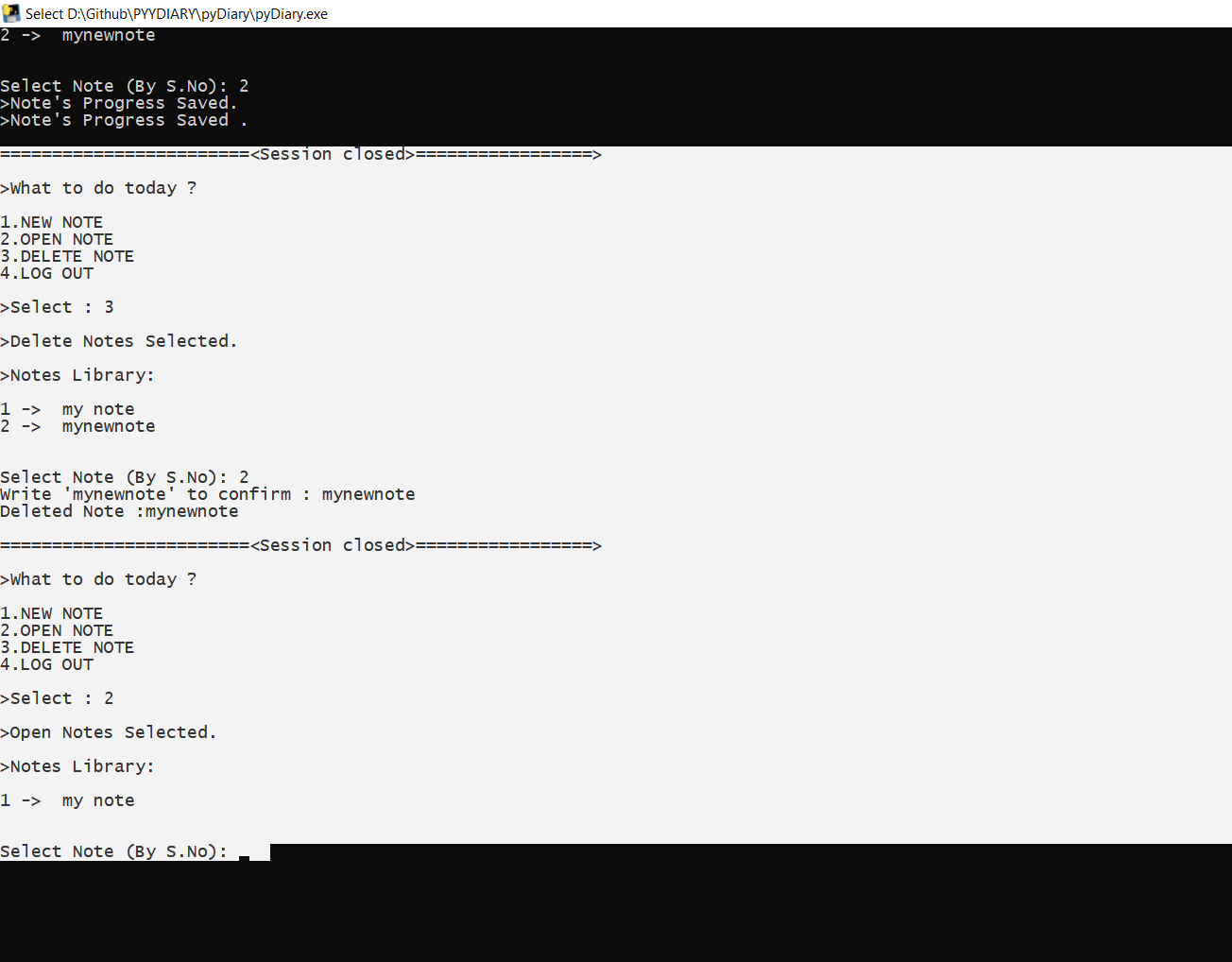


Notes , saved , now enter '2' for viewing all the notes available.



'MyNewNote' is opened in the pydiary editor. Fig. below shows pyDiary's responsiveness to editor hieght change.





Chosed '3' to delete the 'MyNewNote' note and then showing that it is successfully deleted.

# SOFTWARE REQUIREMENTS

**SOFTWARE REQUIREMENTS:**

1. **Windows OS**
2. **Python 2.7/3.2**
3. **XAMPP SERVER or equvivalent sql server**

### INSTALLATION PROCEDURE

**PyDiary :-**

Pre-Requisites :

1. You have to have the following softwares for the successful running of this software; which are

I) Python (Only for the First time), it is downloadable from 'www.python.org'.

II) MySQL (Only for the First time), it is downloadable from 'www.mysql.org'.

Installation :-

1. There will be two folders namely 'PyDiary' and 'Pydiary py file' in the folder 'Source Code'.

2. The folder 'Python Files' will contain the source code of the software in python language. If you are running the software by the 3rd step mentioned below you have to pre install the following modules :-

I) pymysql

II) Tkinter Lib

3. Open the files in any python editors and run it to start and work on the software.

4. The folder 'pyDiary' will contain several files , click 'pyDiary.exe' to run the applicaton.

**BIBLIOGRAPHY**

* ***Book : Computer science With Python - Class XII By : Sumita Arora.***
* ***Website: SQL Documentations :*** https://dev.mysql.com/doc/
* ***Tkinter Documentation***
* ***Pymysql Documentation***

***\*\*\****