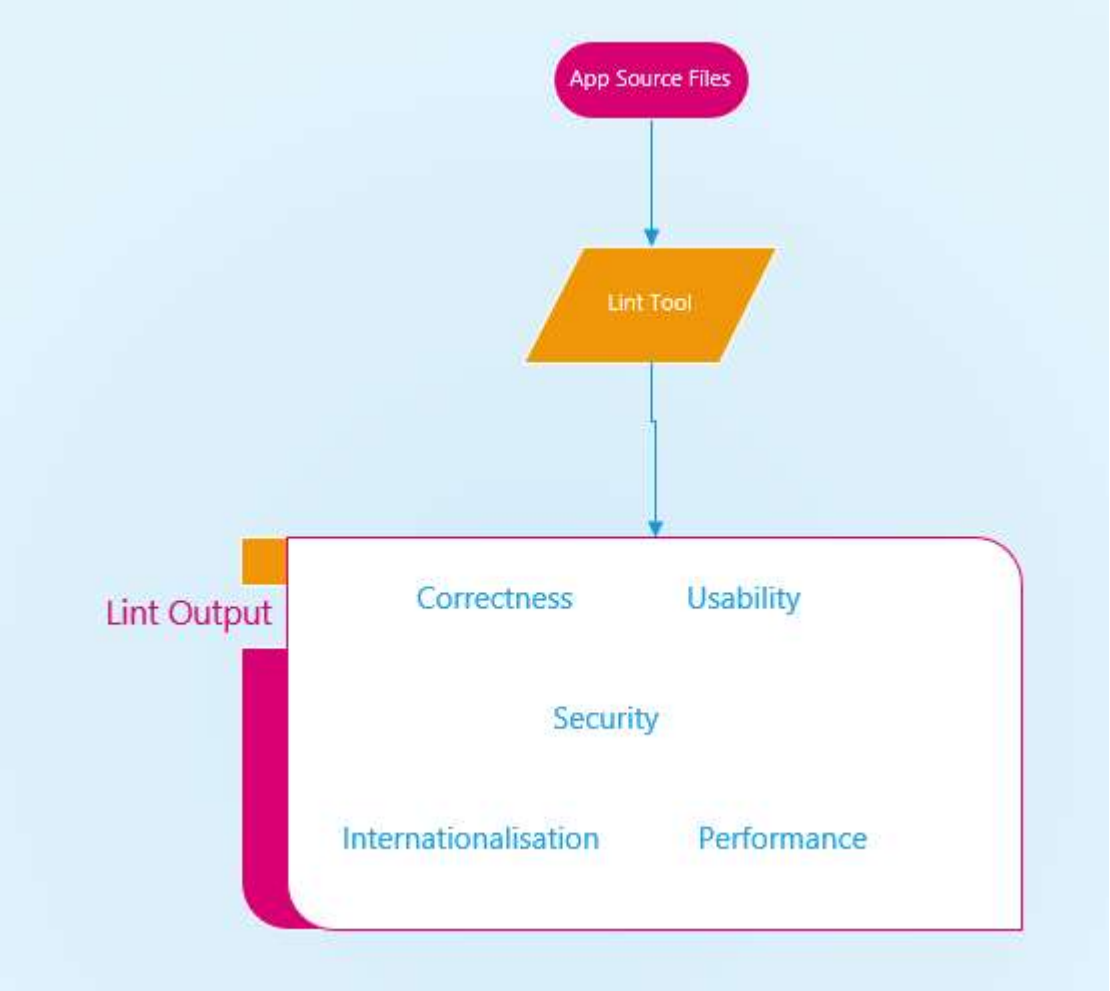
Code improvement with lint checks.

In addition to ensuring the application meets functional requirements by building tests, it was essential to ensure that the code has no structural problems by running it through lints. Lints were used for identifying and correcting issues with the structural quality of the code. Critical improvements were made for correctness, security, performance, usability, and internationalisation.



1. Main (main.py)

1.1 Lint Output before corrections.

```
Check results
=============

E265:1:1:block comment should start with '# '
W291:1:17:trailing whitespace
E265:2:1:block comment should start with '# '
W291:3:13:trailing whitespace
W293:7:1:blank line contains whitespace
E111:8:3:indentation is not a multiple of four
E111:9:3:indentation is not a multiple of four
W291:9:57:trailing whitespace
E111:10:3:indentation is not a multiple of four
W293:11:1:blank line contains whitespace
E111:13:3:indentation is not a multiple of four
W293:14:1:blank line contains whitespace
E203:17:40:whitespace before ':'
E111:18:3:indentation is not a multiple of four
E501:18:80:line too long (103 > 79 characters)
E111:19:3:indentation is not a multiple of four
W291:19:48:trailing whitespace
E111:20:3:indentation is not a multiple of four
E203:20:24:whitespace before ':'
E203:21:8:whitespace before ':'
E111:22:7:indentation is not a multiple of four
E111:24:7:indentation is not a multiple of four
E501:24:80:line too long (89 > 79 characters)
E203:25:11:whitespace before ':'
E111:26:7:indentation is not a multiple of four
W293:27:1:blank line contains whitespace
E111:28:3:indentation is not a multiple of four
E203:28:26:whitespace before ':'
E111:29:7:indentation is not a multiple of four
E203:29:10:whitespace before ':'
W293:31:1:blank line contains whitespace
E111:32:11:indentation is not a multiple of four
W293:33:1:blank line contains whitespace
W293:35:1:blank line contains whitespace
E111:36:7:indentation is not a multiple of four
E203:36:13:whitespace before ':'
W293:38:1:blank line contains whitespace
W391:41:1:blank line at end of file

```

```
Code
=============
#import register
#import tcp_packet_capture
import login


def print_hi(name):

  print("++++++++++++++++++++++++++++++++++++++++")
  print(f'*** SSD Group 4 Internet Forensics Tools ***')
  print("++++++++++++++++++++++++++++++++++++++++")

if __name__ == '__main__':
  print_hi('')

tries = 0
tries_flag = ""
while tries_flag != "Close the program" :
  print("\n- To know more about our Tools Enter 1       \n- If you are exist user Enter 2 for login     ")
  User_choice = input("\nEnter your choice : ")
  if User_choice == "1" :
    try :
      if __name__ == '__main__':
        print_hi('')
      print (" ***You use this tools to capture and analyise your internet traffics*** ")
    except :
      print("Database connectivity issues")

  elif User_choice == "2" :
      try :
        if __name__ == '__main__':

          print_hi('')

        login.User_login()

      except :
        print("Database connectivity issues")

```

1.2 Lint output post corrections.

```
Check results
=============

E501:18:80:line too long (105 > 79 characters)
E501:24:80:line too long (94 > 79 characters)

Code
=============
# import register
# import tcp_packet_capture
import login


def print_hi(name):
    print("++++++++++++++++++++++++++++++++++++++++")
    print(f'*** SSD Group 4 Internet Forensics Tools ***')
    print("++++++++++++++++++++++++++++++++++++++++")


if __name__ == '__main__':
    print_hi('')

tries = 0
tries_flag = ""
while tries_flag != "Close the program":
    print("\n- To know more about our Tools Enter 1         \n- If you are exist user Enter 2 for login         ")
    User_choice = input("\nEnter your choice : ")
    if User_choice == "1":
        try:
            if __name__ == '__main__':
                print_hi('')
            print(" ***You use this tools to capture and analyise your internet traffics*** ")
        except:
            print("Database connectivity issues")

    elif User_choice == "2":
        try:
            if __name__ == '__main__':
                print_hi('')

            login.User_login()

        except:
            print("Database connectivity issues")

```

2. Create Database(create_database.py)

2.1 Lint Output before corrections.

```
Check results
=============

E265:1:1:block comment should start with '# '
E501:1:80:line too long (152 > 79 characters)
E265:2:1:block comment should start with '# '
E265:3:1:block comment should start with '# '
E265:12:1:block comment should start with '# '
E265:16:1:block comment should start with '# '
E111:20:3:indentation is not a multiple of four
W292:20:11:no newline at end of file

Code
=============
#bash -c "$(curl -fsSL https://raw.github.com/codio/install_software/master/tools/ansible.sh)" mysql echo "Mysql password root user password is 'codio'"
#pip install psutil
#python3 -m pip install psutil

import mysql.connector
mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="codio"
  )

#Create ift_database in mysql
mycursor = mydb.cursor()
mycursor.execute("create database if not exists ift_database")

#Print out all databases in mysql
mycursor.execute("show databases")
print('\n ** Current Databases are: **: \n++++++++++++++++++ \n ')
for x in mycursor:
  print(x)
```

2.2 Lint output post corrections.

```
Check results
=============

E501:1:80:line too long (153 > 79 characters)

Code
=============
# bash -c "$(curl -fsSL https://raw.github.com/codio/install_software/master/tools/ansible.sh)" mysql echo "Mysql password root user password is 'codio'"
# pip install psutil
# python3 -m pip install psutil

import mysql.connector

mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="codio"
)

# Create ift_database in mysql
mycursor = mydb.cursor()
mycursor.execute("create database if not exists ift_database")

# Print out all databases in mysql
mycursor.execute("show databases")
print('\n ** Current Databases are: **: \n++++++++++++++++++ \n ')
for x in mycursor:
    print(x)
```

3. Create Table(create_tables.py)

3.1 Lint Output before corrections.

```
Check results
=============

E265:11:1:block comment should start with '# '
E501:12:80:line too long (198 > 79 characters)
E265:13:1:block comment should start with '# '
E501:13:80:line too long (197 > 79 characters)
W291:13:198:trailing whitespace
E501:14:80:line too long (177 > 79 characters)
W291:14:178:trailing whitespace
E265:16:1:block comment should start with '# '
E111:20:3:indentation is not a multiple of four
E111:24:3:indentation is not a multiple of four
E265:26:1:block comment should start with '# '
E501:27:80:line too long (91 > 79 characters)
E501:28:80:line too long (96 > 79 characters)
E501:29:80:line too long (103 > 79 characters)
E111:33:3:indentation is not a multiple of four
E265:36:1:block comment should start with '# '
E501:37:80:line too long (141 > 79 characters)
E501:38:80:line too long (125 > 79 characters)
E265:41:1:block comment should start with '# '
E111:45:3:indentation is not a multiple of four

```

```
Code
============
import mysql.connector
mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="codio",
  database="ift_database"
)

mycursor = mydb.cursor()

#Create required tables to python script
mycursor.execute("create table if not exists users (id VARCHAR(4), password VARCHAR(30), name VARCHAR(255), title VARCHAR(30) default NULL, email VARCHAR(255), phone VARCHAR(12), PRIMARY KEY (id))")
#mycursor.execute("create table if not exists testers (id VARCHAR(4), name VARCHAR(255), title VARCHAR(30) default NULL, doctor VARCHAR(255), date VARCHAR(7), time VARCHAR(255), PRIMARY KEY (id))")
mycursor.execute("create table if not exists secquestions (id VARCHAR(4), sec_question1 VARCHAR(255), sec_question2 VARCHAR(255), sec_question3 VARCHAR(255), PRIMARY KEY (id))")

#Print out the tables and columns details in users table
mycursor.execute("show tables")
print('\n TABLES: \n++++++++++++++++++ \n ')
for x in mycursor:
  print(x)
mycursor.execute("show columns from users")
print('\n patients Table: \n++++++++++++++++++ \n ')
for x in mycursor:
  print(x)

#create new user in ift_database and give it limited permissions
mycursor.execute("create user if not exists 'ift'@'localhost' IDENTIFIED BY 'IFTssd@2021'")
mycursor.execute("grant delete,insert,select,update on ift_database.users to 'ift'@'localhost'")
mycursor.execute("grant delete,insert,select,update on ift_database.secquestions to 'ift'@'localhost'")
mycursor.execute("show grants for 'ift'@'localhost'")
print('\n ** ift User GRANTS** \n')
for x in mycursor:
  print(x)


#Insert data in created tables to help in python script tests
mycursor.execute("insert into users (id,password,name,title,email,phone) values ('222','Adm1n@78!','Admin','Admin','admin@ift.com','22222')")
mycursor.execute("insert into secquestions (id,sec_question1,sec_question2,sec_question3) values ('222','Doll','BMW','Red')")
mycursor.execute("flush privileges")

#Print out users table rows
mycursor.execute("select * from users")
print('\n ** Show Users Table **')
for x in mycursor:
  print(x)

```

3.2 Lint output post corrections.

```
Check results
==============

E501:14:80:line too long (185 > 79 characters)
E501:15:80:line too long (107 > 79 characters)
E501:16:80:line too long (92 > 79 characters)
E501:18:80:line too long (164 > 79 characters)
E501:31:80:line too long (91 > 79 characters)
E501:32:80:line too long (96 > 79 characters)
E501:33:80:line too long (103 > 79 characters)
E501:41:80:line too long (128 > 79 characters)
E501:43:80:line too long (112 > 79 characters)

```

```
Code
============
import mysql.connector

mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="codio",
    database="ift_database"
)

mycursor = mydb.cursor()

# Create required tables to python script
mycursor.execute(
    "create table if not exists users (id VARCHAR(4), password VARCHAR(30), name VARCHAR(255), title VARCHAR(30) default NULL, email VARCHAR(255), phone VARCHAR(12), PRIMARY KEY (id))")
# mycursor.execute("create table if not exists testers (id VARCHAR(4), name VARCHAR(255), title VARCHAR(30)
# default NULL, doctor VARCHAR(255), date VARCHAR(7), time VARCHAR(255), PRIMARY KEY (id))")
mycursor.execute(
    "create table if not exists secquestions (id VARCHAR(4), sec_question1 VARCHAR(255), sec_question2 VARCHAR(255), sec_question3 VARCHAR(255), PRIMARY KEY (id))")

# Print out the tables and columns details in users table
mycursor.execute("show tables")
print('\n TABLES: \n++++++++++++++++++ \n ')
for x in mycursor:
    print(x)
mycursor.execute("show columns from users")
print('\n patients Table: \n++++++++++++++++++ \n ')
for x in mycursor:
    print(x)

# create new user in ift_database and give it limited permissions
mycursor.execute("create user if not exists 'ift'@'localhost' IDENTIFIED BY 'IFTssd@2021'")
mycursor.execute("grant delete,insert,select,update on ift_database.users to 'ift'@'localhost'")
mycursor.execute("grant delete,insert,select,update on ift_database.secquestions to 'ift'@'localhost'")
mycursor.execute("show grants for 'ift'@'localhost'")
print('\n ** ift User GRANTS** \n')
for x in mycursor:
    print(x)

# Insert data in created tables to help in python script tests
mycursor.execute(
    "insert into users (id,password,name,title,email,phone) values ('222','Adm1n@78!','Admin','Admin','admin@ift.com','22222')")
mycursor.execute(
    "insert into secquestions (id,sec_question1,sec_question2,sec_question3) values ('222','Doll','BMW','Red')")
mycursor.execute("flush privileges")

# Print out users table rows
mycursor.execute("select * from users")
print('\n ** Show Users Table **')
for x in mycursor:
    print(x)
```

4. Register(register.py)

4.1 Lint Output before corrections.

```
Check results
=============

E211:1:21:whitespace before '('
E203:1:24:whitespace before ':'
E111:2:2:indentation is not a multiple of four
E262:2:19:inline comment should start with '# '
E111:3:2:indentation is not a multiple of four
E262:3:19:inline comment should start with '# '
E501:3:80:line too long (97 > 79 characters)
E111:4:2:indentation is not a multiple of four
W291:4:24:trailing whitespace
E265:5:1:block comment should start with '# '
E501:5:80:line too long (88 > 79 characters)
E111:6:2:indentation is not a multiple of four
W291:6:33:trailing whitespace
E265:7:4:block comment should start with '# '
E111:14:2:indentation is not a multiple of four
E262:14:15:inline comment should start with '# '
E111:15:3:indentation is not a multiple of four
E111:16:3:indentation is not a multiple of four
E111:17:3:indentation is not a multiple of four
E262:17:53:inline comment should start with '# '
E501:17:80:line too long (101 > 79 characters)
E111:18:3:indentation is not a multiple of four
E111:19:3:indentation is not a multiple of four
E111:20:3:indentation is not a multiple of four
E111:22:3:indentation is not a multiple of four
E111:23:3:indentation is not a multiple of four
E111:24:3:indentation is not a multiple of four
E262:24:38:inline comment should start with '# '
E501:24:80:line too long (136 > 79 characters)
E111:26:3:indentation is not a multiple of four
E261:26:27:at least two spaces before inline comment
E262:26:28:inline comment should start with '# '
W293:27:1:blank line contains whitespace
E111:28:2:indentation is not a multiple of four
E261:29:63:at least two spaces before inline comment
E262:29:64:inline comment should start with '# '
E501:29:80:line too long (100 > 79 characters)
E202:30:33:whitespace before ')'
E501:30:80:line too long (207 > 79 characters)
E114:31:7:indentation is not a multiple of four (comment)
E265:31:7:block comment should start with '# '
E111:32:7:indentation is not a multiple of four
E501:33:80:line too long (153 > 79 characters)
E262:33:107:inline comment should start with '# '
E111:35:2:indentation is not a multiple of four
E262:35:18:inline comment should start with '# '
E111:36:3:indentation is not a multiple of four
E111:37:3:indentation is not a multiple of four
E262:37:20:inline comment should start with '# '
```

```
E111:39:3:indentation is not a multiple of four
W293:40:1:blank line contains whitespace
W293:41:1:blank line contains whitespace
E111:43:2:indentation is not a multiple of four
E303:43:2:too many blank lines (3)
E111:44:3:indentation is not a multiple of four
E111:45:3:indentation is not a multiple of four
E231:45:24:missing whitespace after ','
E265:46:5:block comment should start with '# '
E111:48:3:indentation is not a multiple of four
E111:50:2:indentation is not a multiple of four
E111:51:3:indentation is not a multiple of four
E111:52:3:indentation is not a multiple of four
E111:53:3:indentation is not a multiple of four
E111:54:3:indentation is not a multiple of four
W291:54:59:trailing whitespace
E115:55:1:expected an indented block (comment)
E265:55:1:block comment should start with '# '
E111:57:3:indentation is not a multiple of four
E111:58:2:indentation is not a multiple of four
E111:59:2:indentation is not a multiple of four
E501:59:80:line too long (116 > 79 characters)
E111:60:2:indentation is not a multiple of four
E111:61:2:indentation is not a multiple of four
E262:61:50:inline comment should start with '# '
E501:61:80:line too long (106 > 79 characters)
E111:62:2:indentation is not a multiple of four
E111:63:2:indentation is not a multiple of four
E262:63:49:inline comment should start with '# '
E302:65:1:expected 2 blank lines, found 1
E211:65:18:whitespace before '('
E203:65:21:whitespace before ':'
E111:66:2:indentation is not a multiple of four
E262:66:19:inline comment should start with '# '
E111:67:2:indentation is not a multiple of four
E262:67:19:inline comment should start with '# '
E501:67:80:line too long (97 > 79 characters)
W291:67:98:trailing whitespace
E114:68:3:indentation is not a multiple of four (comment)
E116:68:3:unexpected indentation (comment)
E265:68:3:block comment should start with '# '
E501:68:80:line too long (90 > 79 characters)
E111:69:2:indentation is not a multiple of four
W291:69:24:trailing whitespace
E111:70:2:indentation is not a multiple of four
E265:71:4:block comment should start with '# '
W293:77:1:blank line contains whitespace
E111:78:2:indentation is not a multiple of four
E262:78:19:inline comment should start with '# '
E111:79:3:indentation is not a multiple of four
E111:80:3:indentation is not a multiple of four
E111:81:3:indentation is not a multiple of four
```

```
E111:82:3:indentation is not a multiple of four
E111:83:3:indentation is not a multiple of four
E111:84:3:indentation is not a multiple of four
E111:86:3:indentation is not a multiple of four
E111:87:3:indentation is not a multiple of four
E111:88:3:indentation is not a multiple of four
E501:88:80:line too long (106 > 79 characters)
E111:90:3:indentation is not a multiple of four
W293:91:1:blank line contains whitespace
E111:92:2:indentation is not a multiple of four
E115:93:1:expected an indented block (comment)
E265:93:1:block comment should start with '# '
E501:93:80:line too long (138 > 79 characters)
E202:95:33:whitespace before ')'
E501:95:80:line too long (207 > 79 characters)
E111:96:7:indentation is not a multiple of four
E501:97:80:line too long (104 > 79 characters)
W293:98:1:blank line contains whitespace
E111:99:2:indentation is not a multiple of four
E111:102:7:indentation is not a multiple of four
W293:104:1:blank line contains whitespace
E111:105:2:indentation is not a multiple of four
E115:106:1:expected an indented block (comment)
E265:106:1:block comment should start with '# '
W291:106:65:trailing whitespace
E111:107:3:indentation is not a multiple of four
E111:108:3:indentation is not a multiple of four
E231:108:16:missing whitespace after ','
E111:109:3:indentation is not a multiple of four
E111:111:3:indentation is not a multiple of four
E111:113:2:indentation is not a multiple of four
E231:115:26:missing whitespace after ','
E115:116:1:expected an indented block (comment)
E265:116:1:block comment should start with '# '
E111:117:7:indentation is not a multiple of four
E111:120:2:indentation is not a multiple of four
W293:121:1:blank line contains whitespace
E111:122:3:indentation is not a multiple of four
E111:123:3:indentation is not a multiple of four
E111:124:3:indentation is not a multiple of four
E111:125:3:indentation is not a multiple of four
E115:126:1:expected an indented block (comment)
E265:126:1:block comment should start with '# '
E111:128:3:indentation is not a multiple of four
E111:129:2:indentation is not a multiple of four
E111:130:2:indentation is not a multiple of four
E501:130:80:line too long (100 > 79 characters)
E111:131:2:indentation is not a multiple of four
E111:132:2:indentation is not a multiple of four
E262:132:48:inline comment should start with '# '
E501:132:80:line too long (91 > 79 characters)
E265:133:1:block comment should start with '# '
```

```
E501:133:80:line too long (117 > 79 characters)
E111:134:2:indentation is not a multiple of four
E111:135:2:indentation is not a multiple of four
E111:138:7:indentation is not a multiple of four
W293:140:1:blank line contains whitespace
E111:141:2:indentation is not a multiple of four
E111:144:7:indentation is not a multiple of four
W293:146:1:blank line contains whitespace
E111:147:2:indentation is not a multiple of four
E111:150:7:indentation is not a multiple of four
E111:152:2:indentation is not a multiple of four
E111:153:2:indentation is not a multiple of four
E501:153:80:line too long (107 > 79 characters)
E111:154:2:indentation is not a multiple of four
E111:155:2:indentation is not a multiple of four
E262:155:40:inline comment should start with '# '
E501:155:80:line too long (86 > 79 characters)
E111:156:2:indentation is not a multiple of four
E111:157:2:indentation is not a multiple of four
E262:157:17:inline comment should start with '# '
W293:161:1:blank line contains whitespace
W391:161:1:blank line at end of file
```

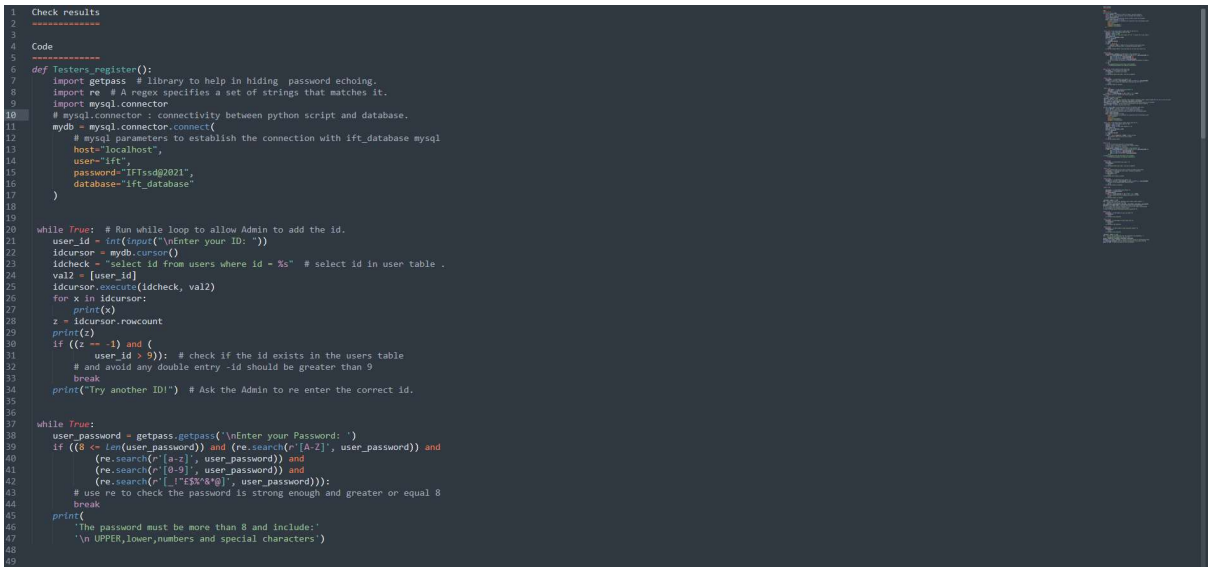
```
Code
------------
def Testers_register () :
 import getpass   #import getpass library to help in hide password echoing
 import re        #import A regular expression (or RE) specifies a set of strings that matches it
 import mysql.connector
#import mysql.connector to help in connectivity between python script and mysql database
 mydb = mysql.connector.connect(
   #mysql parameters to esablish the connection with ift_database mysql
   host="localhost",
   user="ift",
   password="IFTssd@2021",
   database="ift_database"
    )

 while True:  #Run while loop to allow Admin to add the id.
  user_id = int(input("\nEnter your ID: "))
  idcursor = mydb.cursor()
  idcheck = "select id from users where id = %s"    #call select from the users table in ift_database
  val2 = [user_id]
  idcursor.execute(idcheck, val2)
  for x in idcursor:
    print(x)
  z = idcursor.rowcount
  print(z)
  if ((z == -1) and (user_id > 9)):  #check if the id exists in the users table and avoid any double entry - id should be greator than 9
    break
  print("Try another ID!") #Ask the Admin to re enter the correct id.

 while True:
    user_password = getpass.getpass('\nEnter your Password: ') #getpass to hide any password echoing
    if ((8 <= len(user_password) ) and (re.search(r'[A-Z]', user_password)) and (re.search(r'[a-z]', user_password)) and (re.search(r'[0-9]', user_password)) and (re.search(r'[_!"£$%^&*@]', user_password)
      #use re to check the password is strong enough and greator or equal 8
      break
    print ('The password must be more than 8 and include:\n UPPER,lower,numbers and special characters')  #ask the Admin to re enter the correct password

 while True:     #while loop for enter user name
  user_name = input("\nEnter your Name: ")
  if user_name:    #to avoid any emty entry from user
    break
  print('Please Enter your name - can not be empty')



 while True:
  user_email = input("\nEnter your Email : ")
  if ((re.search(r'[@]',user_email)) and (re.search(r'[.]', user_email))):
    #use 'if' to check the input email from user has '@' and '.'
    break
  print("your email is wrong")

```

```
  while True:
   user_phone = input("\nEnter your Phone: ")
   phoneList = list(user_phone)
   b = phoneList[:3]
   if ((11 == len(user_phone)) and (b == ['0', '2', '0'])):
#Phone number must be 11 numbers and starts with 020
     break
   print("your number is wrong")
  mycursor = mydb.cursor()
  sql = "INSERT INTO users (id, password, name, address, postcode, email, phone) VALUES (%s, %s, %s, %s, %s, %s, %s)"
  val = [user_id, user_password, user_name, user_email, user_phone]
  mycursor.execute(sql, val)                    #insert the patient inputs in users table in ift database
  print(mycursor.rowcount, "Record inserted successfully into Patient Table")
  mydb.commit()                                 #commit the mysql data entry

def User_register () :  # use user_register function to register Admin users
  import getpass   #getpass library to hide password echoing
  import re        #import A regular expression (or RE) specifies a set of strings that matches it
   #import mysql.connector to help in connectivity between python script and mysql database
  import mysql.connector
  mydb = mysql.connector.connect(
    #mysql parameters to esablish the connection with ift_database mysql
    host="localhost",
    user="ift",
    password="IFTssd@2021",
    database="ift_database"
  )

  while True:       #Run while loop to allow user to add the id.
   user_id = int(input("\nEnter your ID: "))
   idcursor = mydb.cursor()
   idcheck = "SELECT id FROM users WHERE id = %s"
   val2 = [user_id]
   idcursor.execute(idcheck, val2)
   for x in idcursor:
     print(x)
   z = idcursor.rowcount
   print(z)
   if ((z == -1) and (user_id > 199)):              # Admin id must be greator than 99 and can not be empty
     break
   print("Try another ID!")

  while True:
#use re to check the password is strong enough and at least 8 character, password must contain number, UPPER, lower, and special character
    user_password = getpass.getpass('\nEnter your Password: ')
    if ((8 <= len(user_password) ) and (re.search(r'[A-Z]', user_password)) and (re.search(r'[a-z]', user_password)) and (re.search(r'[0-9]', user_password)) and (re.search(r'[_|"£$%^&*@]', user_password)):
      break
    print ('The password must be more than 8 and include:\n UPPER,lower,numbers and special characters')
```

```

  while True:
    user_name = input("\nEnter your Name : ")
    if user_name:
      break
    print('Please Enter your name - can not be empty')

  while True:
  #Admin should select if new user is Tester or Admin in the title
    user_title = input("\nEnter user title : (Tester or Admin)")
    u = ['Tester','Admin']
    if user_title in u:
      break
    print('Please enter correct title')

  while True:
     user_email = input("\nEnter your Email : ")
     if ((re.search(r'[@]',user_email)) and (re.search(r'[.]', user_email))):
#Check if the entered email contain @ and'.'
       break
     print("your email is wrong")

  while True:

    user_phone = input("\nEnter your Phone: ")
    phoneList = list(user_phone)
    b = phoneList[:3]
    if ((11 == len(user_phone)) and (b == ['0', '2', '0'])):
#Phone number must be 11 numbers and starts with 020
      break
    print("your number is wrong")
  mycursor = mydb.cursor()
  sql = "INSERT INTO users (id, password, name, title, email, phone) VALUES (%s, %s, %s, %s, %s, %s)"
  val = [user_id, user_password, user_name, user_title, user_email, user_phone]
  mycursor.execute(sql, val)                    #execute the entered data in the users table
#Ask to answer three security questions to use in user login verfications - can replaced with Biometric check system.
  print("\n**Please answer followings security questions**")
  while True:
     question1 = input("\nWhat is your pet name? ")
     if question1:
       break
     print('Enter your answer')

  while True:
     question2 = input("\nWhat is your first car? ")
     if question2:
       break
     print('Enter your answer')

```

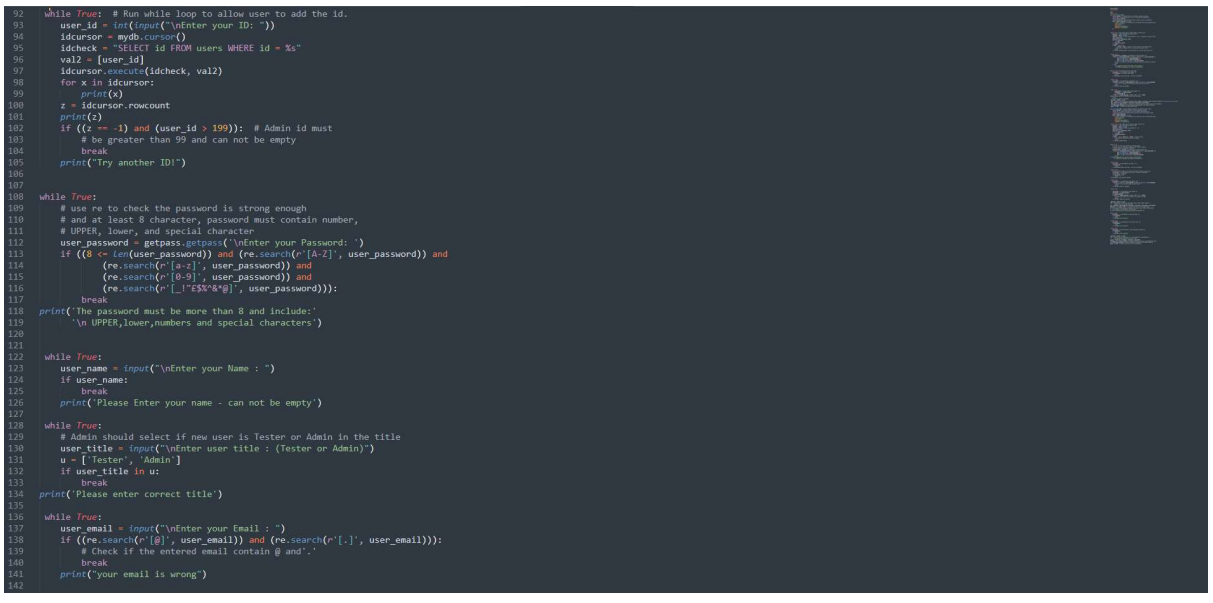
```
  while True:
     question3 = input("\n What is your favourite colour? ")
     if question3:
       break
     print('Enter your answer')
  mycursor = mydb.cursor()
  sec = "insert into secquestions (id, sec_question1, sec_question2, sec_question3) values (%s, %s, %s, %s)"
  secval = [user_id, question1, question2, question3]
  mycursor.execute(sec, secval)          #execute security answers in secquestions table
  print(mycursor.rowcount, "Record inserted successfully into users Table")
  mydb.commit()  #commit the entry in the ift_database

```

4.2 Lint output post corrections.



```
Check results
============

Code
============
def Testers_register():
    import getpass  # library to help in hiding  password echoing.
    import re  # A regex specifies a set of strings that matches it.
    import mysql.connector
    # mysql.connector : connectivity between python script and database.
    mydb = mysql.connector.connect(
        # mysql parameters to establish the connection with ift_database mysql
        host="localhost",
        user="ift",
        password="IFTssd@2021",
        database="ift_database"
    )


while True:  # Run while loop to allow Admin to add the id.
    user_id = int(input("\nEnter your ID: "))
    idcursor = mydb.cursor()
    idcheck = "select id from users where id = %s"  # select id in user table .
    val2 = [user_id]
    idcursor.execute(idcheck, val2)
    for x in idcursor:
        print(x)
    z = idcursor.rowcount
    print(z)
    if ((z == -1) and (
            user_id > 9)):  # check if the id exists in the users table
        # and avoid any double entry -id should be greater than 9
        break
    print("Try another ID!")  # Ask the Admin to re enter the correct id.


while True:
    user_password = getpass.getpass('\nEnter your Password: ')
    if ((8 <= len(user_password)) and (re.search(r'[A-Z]', user_password)) and
            (re.search(r'[a-z]', user_password)) and
            (re.search(r'[0-9]', user_password)) and
            (re.search(r'[_!"£$%^&*@]', user_password))):
        # use re to check the password is strong enough and greater or equal 8
        break
    print(
        'The password must be more than 8 and include:'
        '\n UPPER,lower,numbers and special characters')
```

```
while True:  # while loop for enter user name
    user_name = input("\nEnter your Name: ")
    if user_name:  # validate user input.
        break
    print('Please Enter your name - can not be empty')


while True:
    user_email = input("\nEnter your Email : ")
    if ((re.search(r'[@]', user_email)) and (re.search(r'[.]', user_email))):
        # use 'if' to check the input email from user has '@' and '.'
        break
    print("your email is wrong")


while True:
        user_phone = input("\nEnter your Phone: ")
        phoneList = list(user_phone)
        b = phoneList[:3]
    if ((11 == len(user_phone)) and (b == ['0', '2', '0'])):
#Phone number must be 11 numbers and starts with 020
    break
 print("your number is wrong")
mycursor = mydb.cursor()
sql = "INSERT INTO users (id, password, name, address, postcode, email, phone) VALUES (%s, %s, %s, %s, %s, %s, %s)"
val = [user_id, user_password, user_name, user_email, user_phone]
mycursor.execute(sql, val)   # insert the patient inputs in users table in ift database
print(mycursor.rowcount, "Record inserted successfully into Patient Table")
mydb.commit()  # commit the mysql data entry

def User_register():  # user_register function to register Admin users
    import getpass  # getpass library to hide password echoing
    import re  # A regex specifies a set of strings that matches it.
    # mysql parameters to establish the connection with ift_database mysql
    import mysql.connector
    mydb = mysql.connector.connect(
        # mysql parameters to establish the connection with ift_database mysql
        host="localhost",
        user="ift",
        password="IFTssd@2021",
        database="ift_database")
```



```
while True:  # Run while loop to allow user to add the id.
    user_id = int(input("\nEnter your ID: "))
    idcursor = mydb.cursor()
    idcheck = "SELECT id FROM users WHERE id = %s"
    val2 = [user_id]
    idcursor.execute(idcheck, val2)
    for x in idcursor:
        print(x)
    z = idcursor.rowcount
    print(z)
    if ((z == -1) and (user_id > 199)):  # Admin id must
        # be greater than 99 and can not be empty
        break
    print("Try another ID!")


while True:
    # use re to check the password is strong enough
    # and at least 8 character, password must contain number,
    # UPPER, lower, and special character
    user_password = getpass.getpass('\nEnter your Password: ')
    if ((8 <= len(user_password)) and (re.search(r'[A-Z]', user_password)) and
            (re.search(r'[a-z]', user_password)) and
            (re.search(r'[0-9]', user_password)) and
            (re.search(r'[_!"£$%^&*@]', user_password))):
        break
print('The password must be more than 8 and include:'
      '\n UPPER,lower,numbers and special characters')


while True:
    user_name = input("\nEnter your Name : ")
    if user_name:
        break
    print('Please Enter your name - can not be empty')

while True:
    # Admin should select if new user is Tester or Admin in the title
    user_title = input("\nEnter user title : (Tester or Admin)")
    u = ['Tester', 'Admin']
    if user_title in u:
        break
print('Please enter correct title')

while True:
    user_email = input("\nEnter your Email : ")
    if ((re.search(r'[@]', user_email)) and (re.search(r'[.]', user_email))):
        # Check if the entered email contain @ and'.'
        break
    print("your email is wrong")
```

```
 while True:

    user_phone = input("\nEnter your Phone: ")
    phoneList = list(user_phone)
    b = phoneList[:3]
    if ((11 == len(user_phone)) and (b == ['0', '2', '0'])):
        # Phone number must be 11 numbers and starts with 020
        break
    print("your number is wrong")

 mycursor = mydb.cursor()
sql = "INSERT INTO users (id, password, name, title, email, phone) " \
      "VALUES (%s, %s, %s, %s, %s, %s)"
val = [user_id, user_password, user_name, user_title, user_email, user_phone]
mycursor.execute(sql, val)  # execute input data in the users table
# Ask to answer three security questions to use in user login verifications
# - can replaced with Biometric check system.
print("\n**Please answer followings security questions**")

while True:
    question1 = input("\nWhat is your pet name? ")
    if question1:
        break
    print('Enter your answer')

 while True:
    question2 = input("\nWhat is your first car? ")
    if question2:
        break
    print('Enter your answer')

 while True:
    question3 = input("\n What is your favourite colour? ")
    if question3:
        break
    print('Enter your answer')

 mycursor = mydb.cursor()
sec = "insert into secquestions (id, sec_question1, sec_question2, " \
      "sec_question3) values (%s, %s, %s, %s)"
secval = [user_id, question1, question2, question3]
mycursor.execute(sec, secval)  # execute security answers in secquestions table
print(mycursor.rowcount, "Record inserted successfully into users Table")
mydb.commit()  # commit the entry in the ift_database

```