```
import mysql.connector
import uuid
import hashlib
import os
import re
import getpass
import stdiomask

# Color cods
class colors:
    GREEN = '\033[92m'
    RED = '\033[91m'
    BLUE = '\033[94m'
    BOLD = '\033[1m'

# Database Connection
connectiondb =
mysql.connector.connect(host="localhost",user="root",passwd="codio",database="asmis")
cursordb = connectiondb.cursor()

#Login or register menu
def begin():
    global option

    print(colors.BLUE, "")

    option = input( "login or Register (login, reg, exit): ")

    if(option.lower() == "login" or option.lower() == "reg"):
        menu(option)
    elif(option.lower() == "exit"):
        option = ""
        print(colors.BLUE, "You are successfully logged out.")
    else:
        begin()


def menu(option):

    # Display Menu

    global name
    if(option.lower() == "login"):
      name = input("Enter your Username: ")

      # Display user password input as asterisk
      password = stdiomask.getpass(prompt='Enter your Password: ', mask='*')
      login(name,password)
    else:
      print(colors.BLUE,"Enter your Username and password to register")
      print(colors.GREEN,"For new username use minimum 6 characters & maximum 10
      characters !")
      name = input("Enter Username: ")
      print(colors.GREEN,"Use 8 or more characters with a mix of letters, numbers &
      symbols !")

      # Display user password input as asterisk
      password = stdiomask.getpass(prompt='Enter your Password: ', mask='*')
      register(name,password)

def register(name, password):

    # Manage user registration

    if (name == "" or validate_user_name(name) == False):
        print(colors.RED, "Username does not meet the Username policy requirements.")
        option = "reg"
```

```
65          menu(option)
66      elif (validate_password_strength(password) == False):
67          print(colors.RED,"The password does not meet the password policy requirements.")
68          option = ""
69          begin()
70      elif (validate_user_name_exist(name)):
71          print(colors.RED, "Username was already taken. Please select another")
72          option = "reg"
73          menu(option)
74      else:
75          # Creating a new user & the password will be encrypted before storing.
76
77          salt = os.urandom(32)
78          hashedPassword = hash_password(password, salt)
79          sql = "insert into usertable (username, salt, password, failedcount) values
            (%s, %s, %s, %s)"
80          cursordb.execute(sql, (name, salt, hashedPassword, 0))
81          connectiondb.commit()
82          print(colors.GREEN, "User Registration Successful please login  !!!!!!!!!!!!!!")
83          option = "login"
84          menu(option)
85
86
87  def validate_password_strength(input):
88
89      # Check password strength
90
91      isInvalid = True
92      while isInvalid:
93          if (len(input) < 8 or len(input) > 20):
94              break
95          elif not re.search("[a-z]", input):
96              break
97          elif not re.search("[0-9]", input):
98              break
99          elif not re.search("[A-Z]", input):
100             break
101         elif not re.search("[$#@_!%^&*()-+=:;.,`~]", input):
102             break
103         elif re.search("\s", input):
104             break
105         else:
106             return True
107
108     return False
109
110 def validate_user_name(input):
111
112     # Check username validity _Return True if the username is valid, false otherwise
113
114     isInvalid = True
115     while isInvalid:
116         if (len(input) < 6 or len(input) > 10):
117             break
118         elif re.search(" ", input):
119             break
120
121         else:
122             return True
123
124     return False
125
126 def validate_user_name_exist(name):
127
128     # Check if username already exists
129
130     sql = "select username from usertable where username = %s"
```

```
131         cursordb.execute(sql, [(name)])
132         results = cursordb.fetchall()
133         if results:
134           return True
135 
136     def login(name, password):
137 
138         # Get the user record from database
139         # and check account lock and password validity
140 
141         sql = "select password, salt, failedcount from usertable where username = %s"
142         cursordb.execute(sql, [(name)])
143         results = cursordb.fetchall()
144         if results:
145           for row in results:
146             if (check_password(password, row[0], row[1]) == True):
147               if (check_failed_count(row[2]) == True):
148                 handle_locked_account()
149               else:
150                 handle_authenticated()
151             else:
152                 updateFailedCountSql = "update usertable set failedcount = failedcount + 1
                    where username = %s"
153                 cursordb.execute(updateFailedCountSql, [(name)])
154                 connectiondb.commit()
155 
156                 handle_failed_authentication()
157         else:
158           handle_failed_authentication()
159 
160     def handle_authenticated():
161 
162         # Manage successfull login
163 
164         print(colors.GREEN, "Successfully logged in")
165         logOutOption = input("Do you want to log out? (y/n)")
166         if (logOutOption.lower() == "y"):
167             begin()
168         else:
169             handle_authenticated()
170 
171     def check_password(enteredPassword, dbPassword, salt):
172 
173         # Verify user password for login
174         # Return True if the password is valid, false otherwise
175 
176         hashedPassword = hash_password(enteredPassword, salt)
177 
178         if hashedPassword == dbPassword:
179            return True
180         else:
181             return False
182 
183     def check_failed_count(failedCount):
184 
185         # Check if account is locked
186         # Return True if account exceeds the fail count, false otherwise
187 
188         if (failedCount >= 3):
189             return True
190         else:
191             return False
192 
193     def handle_failed_authentication():
194 
195         # Manage failed login
196 
```

```
197         print(colors.RED,"Wrong Username or Password !!!!!!!")
198         option = ""
199         begin()
200 
201     def handle_locked_account():
202 
203         # Display failed login
204 
205         print(colors.RED,"Your Account Has Been Locked Out Please Contact Administrator !!!")
206         option = ""
207         begin()
208 
209     def hash_password(password, salt):
210 
211         # Encrypt the password and return the sha digest
212 
213         return hashlib.pbkdf2_hmac('sha256', password.encode('utf-8'), salt, 100000)
214 
215 
216     print(colors.BLUE,colors.BOLD, "Welcome to the ASMIS System")
217     begin()
218 
219 
220 
```