Problem 1:

a) I chose RBAC as the access control model to be used in the development of my prototype because the access control policy determines the user's functionality based on their role. For example, Administrators can do everything that employees can, but they encompass the only role who can modify a patient's profile. Therefore, it should be deemed that the role of an Administrator is not equivalent to the role of an employee, with respect to access privileges. The RBAC model will permit my consulting firm to authorize the validated user(s) according to their designated role. RBAC is also a great choice since we are dealing with important data, we do not want the wrong user to have access to modifying a patient's history or suggesting a diagnosis. Therefore, we can ensure data is managed properly with the RBAC model.

b) My consulting firm chose an access control matrix as the representation for our RBAC access control policy control model because the ACM allows us to assign role(s) to each user. The ACM also encapsulates the Principle of Least Privilege, meaning no user will have access to anything that is not necessary for their role within the system.

c)

		Role					
		Radiologist	Physician	Nurse	Patient	Administrator	Technical Support
User	Howard Linkler	y					
	Veronica Perez	y					
	Winston Callahan		y				
	Leslie Stewart		y				
	Kelsey Chang			y			
	James Preston			y			
	Rahul Garza				y		
	Arjun Singh				y		
	Heather Anderson					y	
	Keikilana Kapahu					y	
	Harold Zakara						y
	Malina Romanova						y

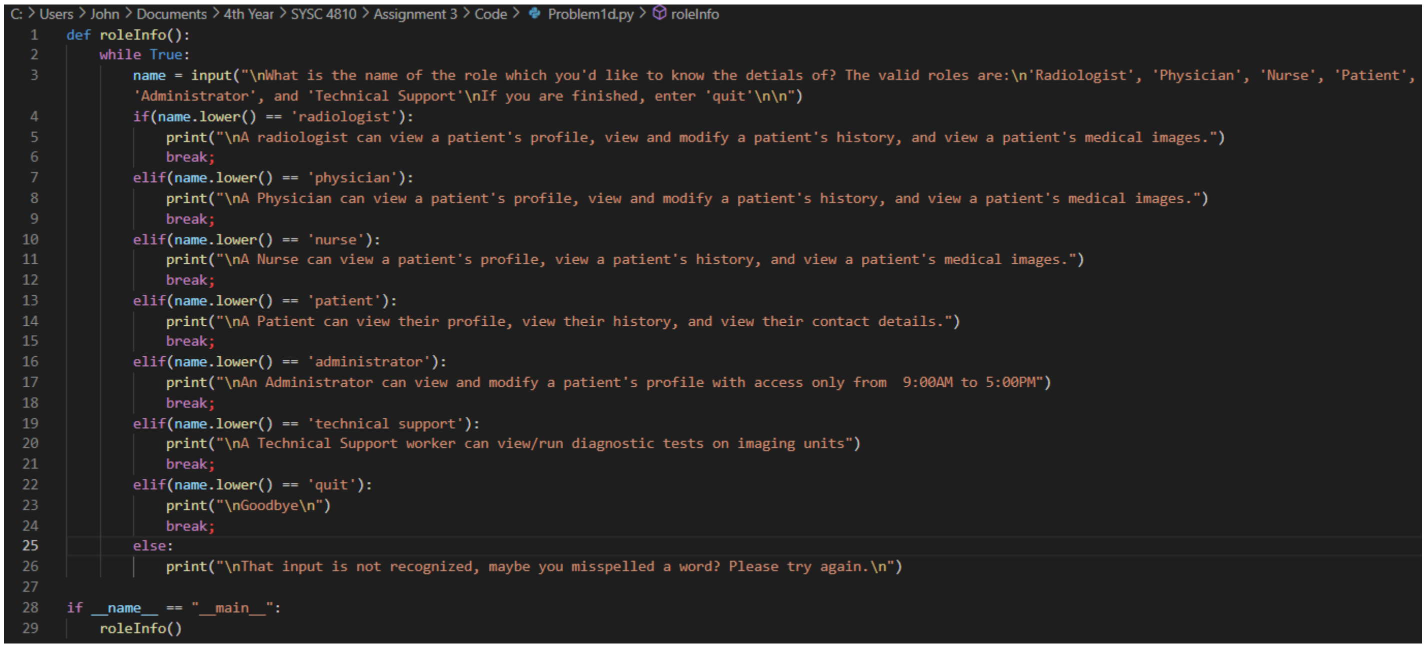
Figure 1 - Y indicates that the User is authorized for the role

d)

	Patient profile	Patient history	Physician contact details	Patient medical images	Imaging Units
Radiologist	view	view, modify		view	
Physician	view	view, modify		view	
Nurse	view	view		view	
Patient	view	view	view		
Administrator	view, modify *(restricted)*				
Technical Support					view

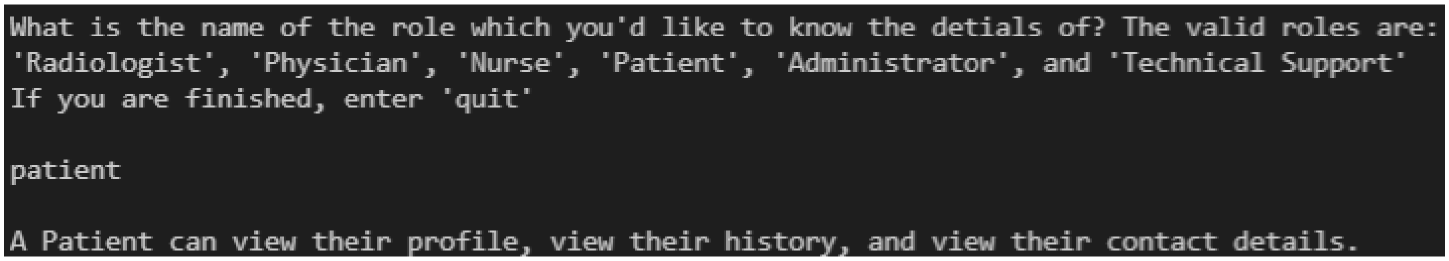
Assumptions:

- Administrators are ***restricted*** because they can only access the system during business hours from 9:00AM to 5:00PM.
- While Administrators can modify a patient's profile, it is not explicitly stated that they can view patient's history, details, medical images.
- A radiologist and physician can (either directly or indirectly) modify a patient's history by suggesting a diagnosis which gets written into a patient's history
- A physician can also (either directly or indirectly) modify a patient's history by prescribing a treatment which gets written into a patient's history.


C: > Users > John > Documents > 4th Year > SYSC 4810 > Assignment 3 > Code > Problem1d.py > roleInfo

```python
def roleInfo():
    while True:
        name = input("\nWhat is the name of the role which you'd like to know the detials of? The valid roles are:\n'Radiologist', 'Physician', 'Nurse', 'Patient', 
        'Administrator', and 'Technical Support'\nIf you are finished, enter 'quit'\n\n")
        if(name.lower() == 'radiologist'):
            print("\nA radiologist can view a patient's profile, view and modify a patient's history, and view a patient's medical images.")
            break;
        elif(name.lower() == 'physician'):
            print("\nA Physician can view a patient's profile, view and modify a patient's history, and view a patient's medical images.")
            break;
        elif(name.lower() == 'nurse'):
            print("\nA Nurse can view a patient's profile, view a patient's history, and view a patient's medical images.")
            break;
        elif(name.lower() == 'patient'):
            print("\nA Patient can view their profile, view their history, and view their contact details.")
            break;
        elif(name.lower() == 'administrator'):
            print("\nAn Administrator can view and modify a patient's profile with access only from  9:00AM to 5:00PM")
            break;
        elif(name.lower() == 'technical support'):
            print("\nA Technical Support worker can view/run diagnostic tests on imaging units")
            break;
        elif(name.lower() == 'quit'):
            print("\nGoodbye\n")
            break;
        else:
            print("\nThat input is not recognized, maybe you misspelled a word? Please try again.\n")

if __name__ == "__main__":
    roleInfo()
```

Figure 2 - Problem1d.py



```
What is the name of the role which you'd like to know the detials of? The valid roles are:
'Radiologist', 'Physician', 'Nurse', 'Patient', 'Administrator', and 'Technical Support'
If you are finished, enter 'quit'

patient

A Patient can view their profile, view their history, and view their contact details.
```

I took a straightforward approach with implementing my ACM, I wanted a simplistic yet interactive experience. I designed the ACM using the two points below:

- Prompt user to enter the name of their role to see what privileges each role entails.
- After user enters a role, the system prints the privileges assigned to that role (please note that entering a role doesn't grant the user that role, it only shows them what privileges a person with that role has).

e) I tested the ACM by creating a new user with a specified role, logging into the system, and verifying if the user privilege(s) according to their role were the same as the privilege(s) shown in the ACM.

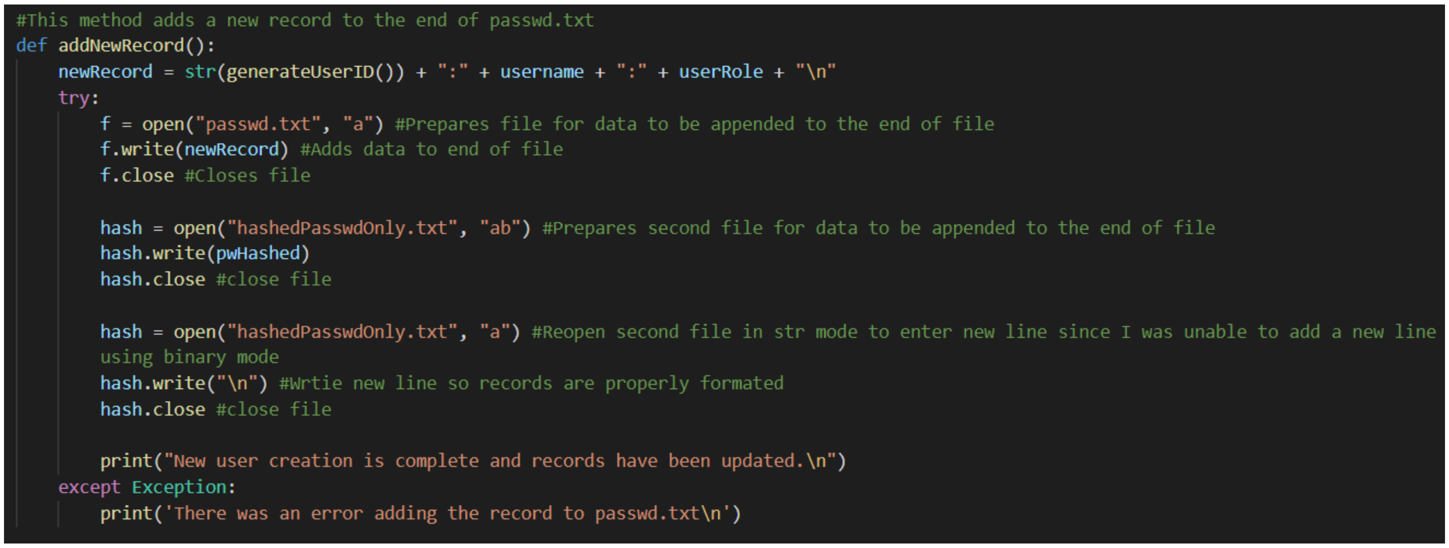
Problem 2:

a) Although it was stated that MD5 is amongst the most popular choice, it isn't suitable for this system because it has been found to suffer from extensive vulnerabilities. Therefore, the specific hashing algorithm that my consulting firm will employ in our password file mechanism is Bcrypt. Bcrypt was introduced in 1999, it is a well-known and established hashing algorithm. It is however not perfect; it is more susceptible to dictionary attacks than other algorithms like Scrypt. Bcrypt is still a very ideal choice because it has a balance of performance and security, although it is old, it still stands the test of time. We will also implement a password checker which will help cover some dictionary words that Bcrypt may be susceptible to. Bcrypt has a function gensalt() which will auto generate a salt. Bcrypt supports a maximum hash length of 56 bytes, but the auto generated hash for Medview users will vary and likely less due to the password length being 8-12 characters. The method, gensalt(), will take care of the salt length, salt generation requirements my firm must fulfil, and will therefore be one less requirement for us to worry about.

Assumption/Notes: The PDF never restricted us from using automated methods like gensalt(), therefore I opted for it.

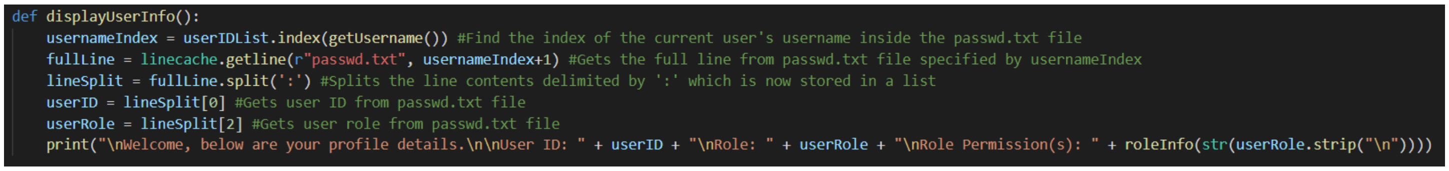
b) The structure of each record in our password file is userID:username:hash:roleName where,
userID is a random unique user identification number.
username is typically the user's first and last name combined in lower case.
hash is the hashed form of the password
roleName is a string identifier corresponding to the role of the user, which provides them the respective privilege(s).

c) My firm implemented our database using two password files, passwd.txt and hashedPasswdOnly.txt *(please read explanation after Problem 5 summary titled* ***<u>Final Assumptions/Notes/Thoughts/Issues</u>*** *as to why there are two .txt files instead of one)*. When a new user inputs the necessary details, the database saves their userID:username:roleName in passwd.txt and their hash in hashedPasswdOnly.txt. The password is hashed using bcrypt algorithm.



```
#This method adds a new record to the end of passwd.txt
def addNewRecord():
    newRecord = str(generateUserID()) + ":" + username + ":" + userRole + "\n"
    try:
        f = open("passwd.txt", "a") #Prepares file for data to be appended to the end of file
        f.write(newRecord) #Adds data to end of file
        f.close #Closes file

        hash = open("hashedPasswdOnly.txt", "ab") #Prepares second file for data to be appended to the end of file
        hash.write(pwHashed)
        hash.close #close file

        hash = open("hashedPasswdOnly.txt", "a") #Reopen second file in str mode to enter new line since I was unable to add a new line
        using binary mode
        hash.write("\n") #Wrtie new line so records are properly formated
        hash.close #close file

        print("New user creation is complete and records have been updated.\n")
    except Exception:
        print('There was an error adding the record to passwd.txt\n')
```

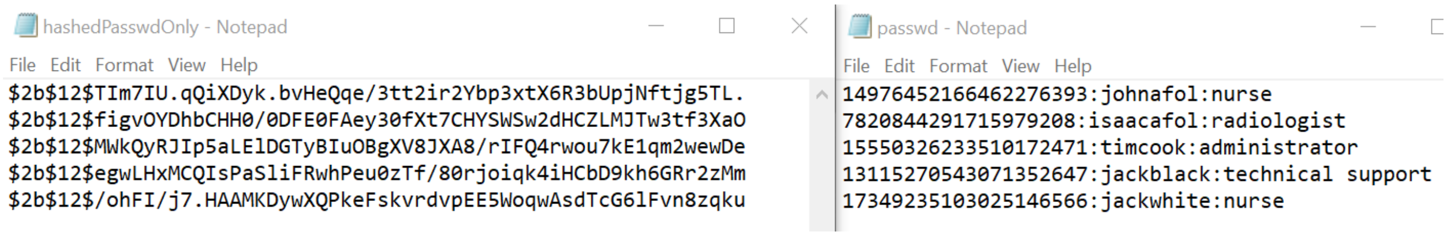
Figure 3 - Adding New Record to txt Files



```
def displayUserInfo():
    usernameIndex = userIDList.index(getUsername()) #Find the index of the current user's username inside the passwd.txt file
    fullLine = linecache.getline(r"passwd.txt", usernameIndex+1) #Gets the full line from passwd.txt file specified by usernameIndex
    lineSplit = fullLine.split(':') #Splits the line contents delimited by ':' which is now stored in a list
    userID = lineSplit[0] #Gets user ID from passwd.txt file
    userRole = lineSplit[2] #Gets user role from passwd.txt file
    print("\nWelcome, below are your profile details.\n\nUser ID: " + userID + "\nRole: " + userRole + "\nRole Permission(s): " + roleInfo(str(userRole.strip("\n"))))
```

Figure 4 - Retrieving Record from File

I tested the password file mechanism by creating a new user, in which the new user's password is stored in the txt files. I then opened the files after user creation to verify that the password structure was as expected, and I confirmed everything functioned properly.

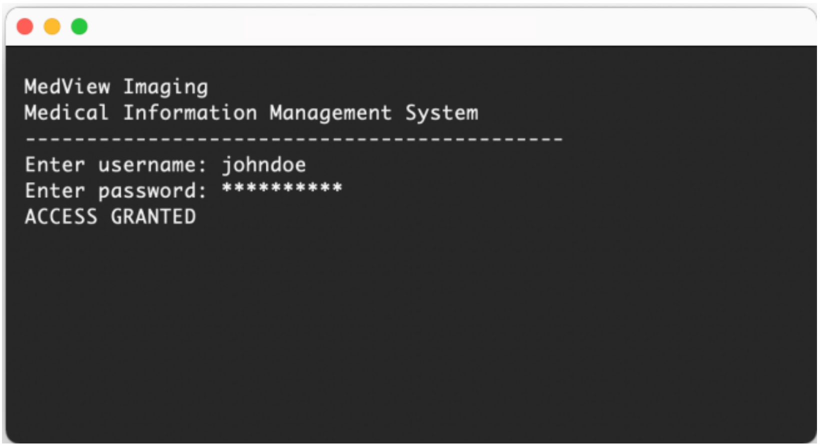

hashedPasswdOnly - Notepad
File Edit Format View Help
passwd - Notepad
File Edit Format View Help

```
$2b$12$TIm7IU.qQiXDyk.bvHeQqe/3tt2ir2Ybp3xtX6R3bUpjNftjg5TL.
$2b$12$figvOYDhbCHH0/0DFE0FAey30fXt7CHYSWSw2dHCZLMJTw3tf3XaO
$2b$12$MWkQyRJIp5aLElDGTyBIuOBgXV8JXA8/rIFQ4rwou7kE1qm2wewDe
$2b$12$egwLHxMCQIsPaSliFRwhPeu0zTf/80rjoiqk4iHCbD9kh6GRr2zMm
$2b$12$/ohFI/j7.HAAMKDywXQPkeFskvrdvpEE5WoqwAsdTcG6lFvn8zqku
```

```
14976452166462276393:johnafol:nurse
7820844291715979208:isaacafol:radiologist
15550326233510172471:timcook:administrator
13115270543071352647:jackblack:technical support
17349235103025146566:jackwhite:nurse
```

Problem 3:

a) My firm modeled our user interface to look very similar like the sample sent by our client seen in Figure 5 below. In order to enroll a user, we created a welcome page which would ask a user if they wished to log into an existing account or create a new one. We designed a prototype (seen in Figure 6) before implementing the design. In order to a enroll user, the system must ask them if they have an existing account or if they would like to create one, my firm did exactly this.



```
MedView Imaging
Medical Information Management System
--------------------------------------------
Enter username: johndoe
Enter password: **********
ACCESS GRANTED
```

Figure 5 - Sample UI provided

Please note that text highlighted is what we want to do, text in white is what the user will see on console, text entered in red is user input and text in gold is the step the program will repeat after user input.

(1) Upon entering the system display:

Welcome to Medview Imaging

Medical Information Management System

Do you have an existing profile?

Enter 'y' for yes or 'n' for no

(2) If user enters 'n', then walk them through

creating an account in order to enrol them:

(in this case we are only concerned about ‘n’ because that is the only scenario which a user would be enrolled)

Medview Imaging

New User Profile:

(2a) ask user for name, (any input will be accepted) Please enter your first name: User enters first name

(2b) ask user for username, (any input will be accepted as long as username is not already taken) Please enter the profile username of your preference: User enters username of preference

(2c) if username is taken, ask user to enter a different username (repeat 2b)

(2d) ask user for password Hi firstName, please create a new password for your account.

Please note that:

Passwords must be least 8-12 characters in length

Password must include at least:

– one upper-case letter;

– one lower-case letter;

– one numerical digit, and

– one special character from the set: {!, @, #, $, %, ?, ∗}

User enters password

(2e) If user inputted password is satisfactory, prompt user to confirm:

User re-enters password

(2f) If user inputted password is not satisfactory, prompt user to start over:

Hi firstName, please create a new password for your account.

Please note that:

Passwords must be least 8-12….

......– one special character from the set: {!, @, #, $, %, ?, ∗}

(2g) If user confirmed password doesn't match original, prompt user to start over:

Hi firstName, please create a new password for your account.

Please note that:

Passwords must be least 8-12….

......– one special character from the set: {!, @, #, $, %, ?, ∗}

(2h) If user confirmed password matches the original, then enrolling process is complete, proceed to save info to file:

Skip to step (3a)

User enters password

(3a) Once user enters the satisfactory required info, then ask for role before saving
their info into .txt file database.

Goes to step 4

(3b) Otherwise, there was an unexpected input. Prompt user to try again or start over:

Scenario 1 (Passwords don't match):

The password you entered does not match the original, please try again. Enter 'exit' if you'd like to start over and create a different password

User enters 'exit'

Repeat step 2

Scenario 2 (Password is not strong enough):

Sorry, the password you entered does not meet the requirements. Please try again.

Repeat step 2d

Scenario 3 (Username is taken):

Sorry, the username you entered already taken. Please try again.

Repeat step 2b

(4) Password is collected. Now ask the user for their role before saving info to file:

What is the name of the role which you will be taking on here at Medview Imaging?

The valid roles are:

'Radiologist', 'Physician', 'Nurse', 'Patient', 'Administrator', and 'Technical Support'

(4a) User enters valid role:

User enters role, e.g: nurse

Goes to step 5

(4b) User enters unexpected input:

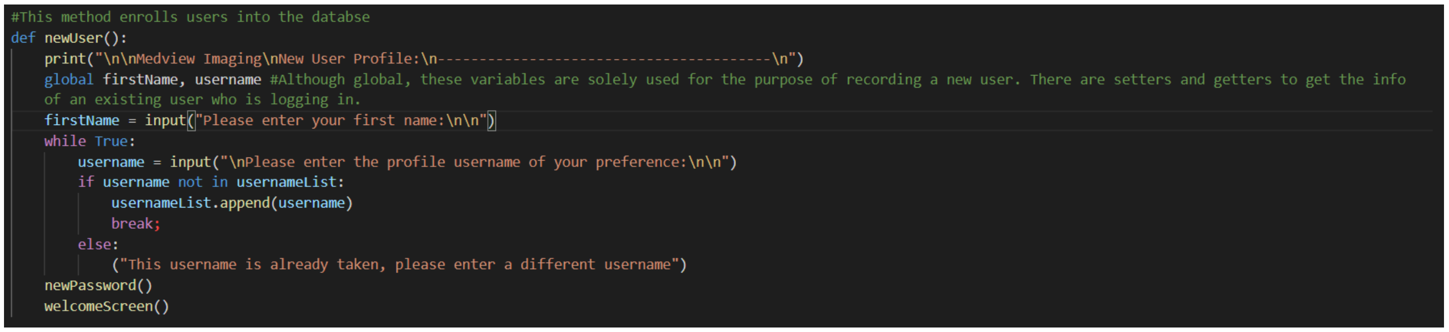
User enters role, e.g: doctor

That input is not recognized, maybe you misspelled a word? Please try again.

Repeat step 4

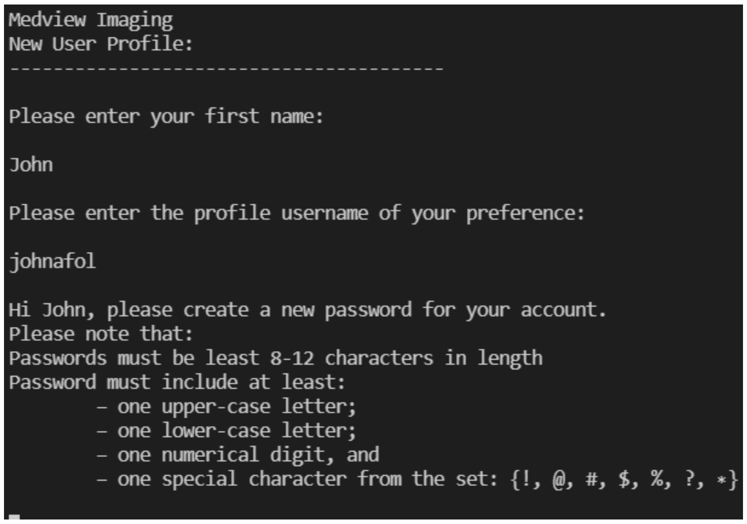
(5) Data is saved to files:

New user creation is complete, and records have been updated.



```
#This method enrolls users into the databse
def newUser():
    print("\n\nMedview Imaging\nNew User Profile:\n----------------------------------------\n")
    global firstName, username #Although global, these variables are solely used for the purpose of recording a new user. There are setters and getters to get the info
    of an existing user who is logging in.
    firstName = input("Please enter your first name:\n\n")
    while True:
        username = input("\nPlease enter the profile username of your preference:\n\n")
        if username not in usernameList:
            usernameList.append(username)
            break;
        else:
            ("This username is already taken, please enter a different username")
    newPassword()
    welcomeScreen()
```

Figure 6 - Method which enrolls new user



```
Medview Imaging
New User Profile:
----------------------------------------

Please enter your first name:

John

Please enter the profile username of your preference:

johnafol

Hi John, please create a new password for your account.
Please note that:
Passwords must be least 8-12 characters in length
Password must include at least:
        - one upper-case letter;
        - one lower-case letter;
        - one numerical digit, and
        - one special character from the set: {!, @, #, $, %, ?, *}
```

Figure 7 - Output of enroll user function

b) To design my password checker, my firm gathered the top 200 most common passwords used online. We then iterated through the list to check if the new user's password was found amongst the list of commonly used passwords.

Pseudocode:

- Pass user's password as a parameter to password checker method
- Create a list of top 200 most used passwords.
- Run a for loop to iterate through list.
- Include an if statement to check if pw passed as parameter is equal to password in list.
- Return true if found or false if not found.

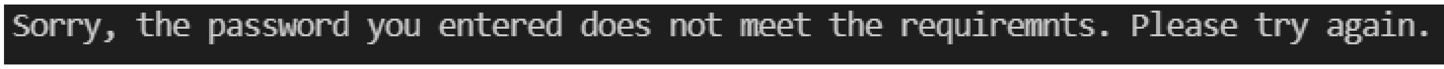
I chose this method because if a dictionary attack were to occur, it is highly likely that words inside the commonly used password list would be ran. Therefore, it is best to prepare for this scenario to the likelihood of attack success.

c) Pseudocode:

- Create a Boolean method, called isPasswordWeak which is passed the new user's password as a parameter

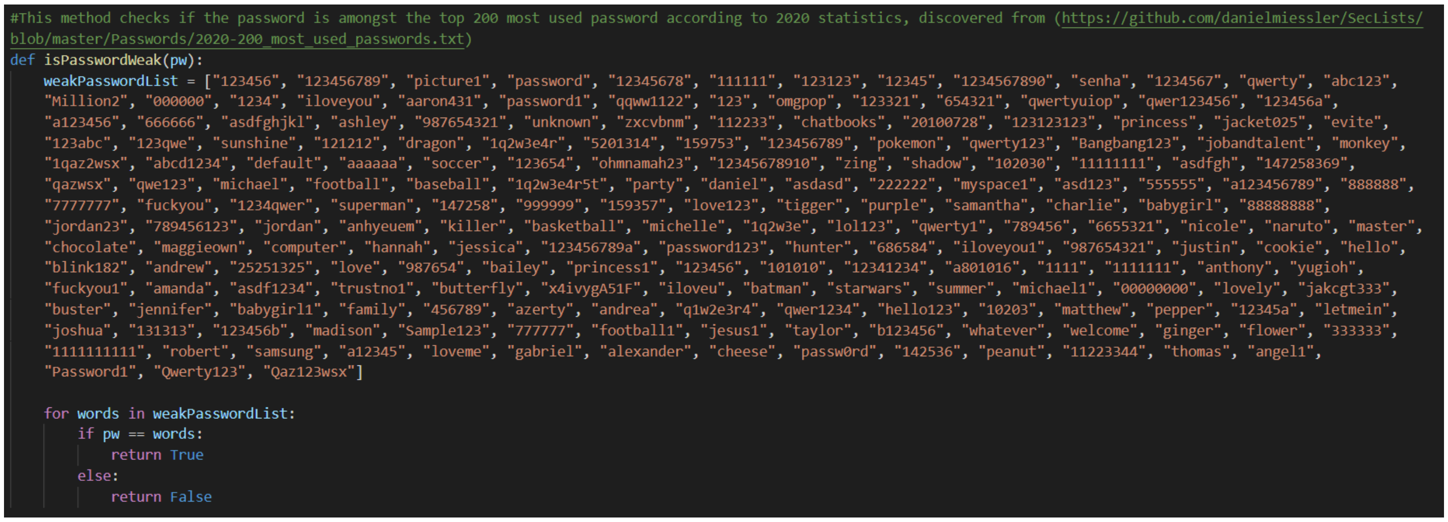
- In isPasswordWeak we will store commonly used passwords (weak passwords checked during dictionary attacks) into a list.
- Scan through the list to see if the user’s password is in that list
- Return true if the user’s password is found in the list or false if it is not found in the list.
- Create a second Boolean method, called isLoginPasswordValid, which is passed the new user’s password as a parameter
- In newPassword(), we will check to see if the new user’s password contains one upper-case letter; one lower-case letter; one numerical digit, and one special character from the set: {!, @, #, $, %, ?, ∗}
- If so, the method will return true, if not, the method returns false.

I tested these two methods by creating a new user and entering passwords from the list of weak passwords which I found online, as well as passwords that were not up to the criteria of which client asked for, and the system rejected the passwords as expected.



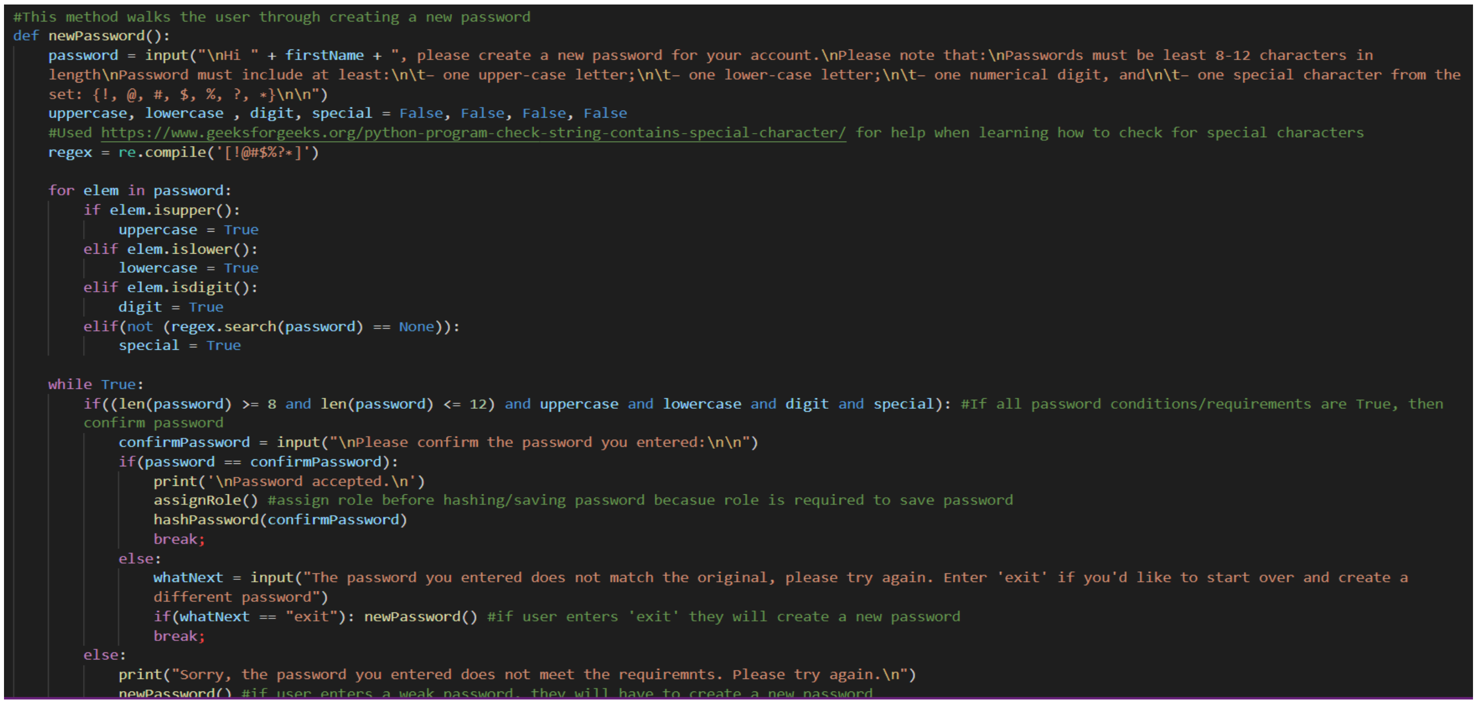
```
Sorry, the password you entered does not meet the requiremnts. Please try again.
```

d)



```python
#This method checks if the password is amongst the top 200 most used password according to 2020 statistics, discovered from (https://github.com/danielmiessler/SecLists/
blob/master/Passwords/2020-200_most_used_passwords.txt)
def isPasswordWeak(pw):
    weakPasswordList = ["123456", "123456789", "picture1", "password", "12345678", "111111", "123123", "12345", "1234567890", "senha", "1234567", "qwerty", "abc123",
    "Million2", "000000", "1234", "iloveyou", "aaron431", "password1", "qqww1122", "123", "omgpop", "123321", "654321", "qwertyuiop", "qwer123456", "123456a",
    "a123456", "666666", "asdfghjkl", "ashley", "987654321", "unknown", "zxcvbnm", "112233", "chatbooks", "20100728", "123123123", "princess", "jacket025", "evite",
    "123abc", "123qwe", "sunshine", "121212", "dragon", "1q2w3e4r", "5201314", "159753", "123456789", "pokemon", "qwerty123", "Bangbang123", "jobandtalent", "monkey",
    "1qaz2wsx", "abcd1234", "default", "aaaaaa", "soccer", "123654", "ohmnamah23", "12345678910", "zing", "shadow", "102030", "11111111", "asdfgh", "147258369",
    "qazwsx", "qwe123", "michael", "football", "baseball", "1q2w3e4r5t", "party", "daniel", "asdasd", "222222", "myspace1", "asd123", "555555", "a123456789", "888888",
    "7777777", "fuckyou", "1234qwer", "superman", "147258", "999999", "159357", "love123", "tigger", "purple", "samantha", "charlie", "babygirl", "88888888",
    "jordan23", "789456123", "jordan", "anhyeuem", "killer", "basketball", "michelle", "1q2w3e", "lol123", "qwerty1", "789456", "6655321", "nicole", "naruto", "master",
    "chocolate", "maggieown", "computer", "hannah", "jessica", "123456789a", "password123", "hunter", "686584", "iloveyou1", "987654321", "justin", "cookie", "hello",
    "blink182", "andrew", "25251325", "love", "987654", "bailey", "princess1", "123456", "101010", "12341234", "a801016", "1111", "1111111", "anthony", "yugioh",
    "fuckyou1", "amanda", "asdf1234", "trustno1", "butterfly", "x4ivygA51F", "iloveu", "batman", "starwars", "summer", "michael1", "00000000", "lovely", "jakcgt333",
    "buster", "jennifer", "babygirl1", "family", "456789", "azerty", "andrea", "q1w2e3r4", "qwer1234", "hello123", "10203", "matthew", "pepper", "12345a", "letmein",
    "joshua", "131313", "123456b", "madison", "Sample123", "777777", "football1", "jesus1", "taylor", "b123456", "whatever", "welcome", "ginger", "flower", "333333",
    "1111111111", "robert", "samsung", "a12345", "loveme", "gabriel", "alexander", "cheese", "passw0rd", "142536", "peanut", "11223344", "thomas", "angel1",
    "Password1", "Qwerty123", "Qaz123wsx"]

    for words in weakPasswordList:
        if pw == words:
            return True
        else:
            return False
```

Figure 8 - isPasswordWeak() checks for commonly used passwords



```python
#This method walks the user through creating a new password
def newPassword():
    password = input("\nHi " + firstName + ", please create a new password for your account.\nPlease note that:\nPasswords must be least 8-12 characters in
    length\nPassword must include at least:\n\t- one upper-case letter;\n\t- one lower-case letter;\n\t- one numerical digit, and\n\t- one special character from the
    set: {!, @, #, $, %, ?, *}\n\n")
    uppercase, lowercase , digit, special = False, False, False, False
    #Used https://www.geeksforgeeks.org/python-program-check-string-contains-special-character/ for help when learning how to check for special characters
    regex = re.compile('[!@#$%?*]')

    for elem in password:
        if elem.isupper():
            uppercase = True
        elif elem.islower():
            lowercase = True
        elif elem.isdigit():
            digit = True
        elif(not (regex.search(password) == None)):
            special = True

    while True:
        if((len(password) >= 8 and len(password) <= 12) and uppercase and lowercase and digit and special): #If all password conditions/requirements are True, then
        confirm password
            confirmPassword = input("\nPlease confirm the password you entered:\n\n")
            if(password == confirmPassword):
                print('\nPassword accepted.\n')
                assignRole() #assign role before hashing/saving password becasue role is required to save password
                hashPassword(confirmPassword)
                break;
            else:
                whatNext = input("The password you entered does not match the original, please try again. Enter 'exit' if you'd like to start over and create a
                different password")
                if(whatNext == "exit"): newPassword() #if user enters 'exit' they will create a new password
                break;
        else:
            print("Sorry, the password you entered does not meet the requiremnts. Please try again.\n")
            newPassword() #if user enters a weak password, they will have to create a new password
```

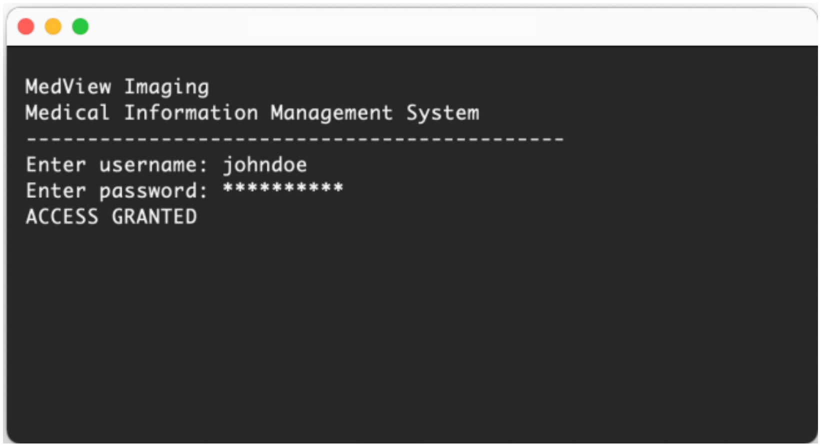
Figure 9 - newPassword() checks to see if the password which a user chooses is compliant with Medview standards

I implemented this design by following my pseudocode and converting each line of English into a lines of code.

Problem 4:

Please note that text highlighted is what we want to do, text in white is what the user will see on console, text entered in red is user input and text in gold is the step the program will repeat after user input.

a) My firm designed a login page visually similar to figure 5 below. After a user chooses to log in, they will have a couple options:



```
MedView Imaging
Medical Information Management System
----------------------------------------
Enter username: johndoe
Enter password: **********
ACCESS GRANTED
```

Figure 10 - Sample UI provided

(1) Upon choosing to login, the system will display:

Medview Imaging

Medical Information Management System

Enter username: User enters 'johnafol'

(1a) If the user name is not found in the .txt file database, the system will prompt the user that the username doesn't exist and ask what they want to do next:

That username doesn't exist, perhaps you misspelt it?

Enter 'y' to try again

Enter 'n' to create a new profile

Enter 'q' to exit back to the main menu

User chooses what to do next

(1a) If the username is found in the .txt file database, the system will prompt the user for their password

Medview Imaging

Medical Information Management System

Enter username: User enters 'johnafol'

Enter password: User enters ‘a1!Aaaaa’

(1c) If password is same as password in .txt file, then allow access and display user info

```
ACCESS GRANTED

Welcome, below are your profile details.

User ID: 14976452166462276393
Role: nurse

Role Permission(s):  You can view a patient's profile, view a patient's history, and view a patient's medical images.
 Please enter your first name:
```

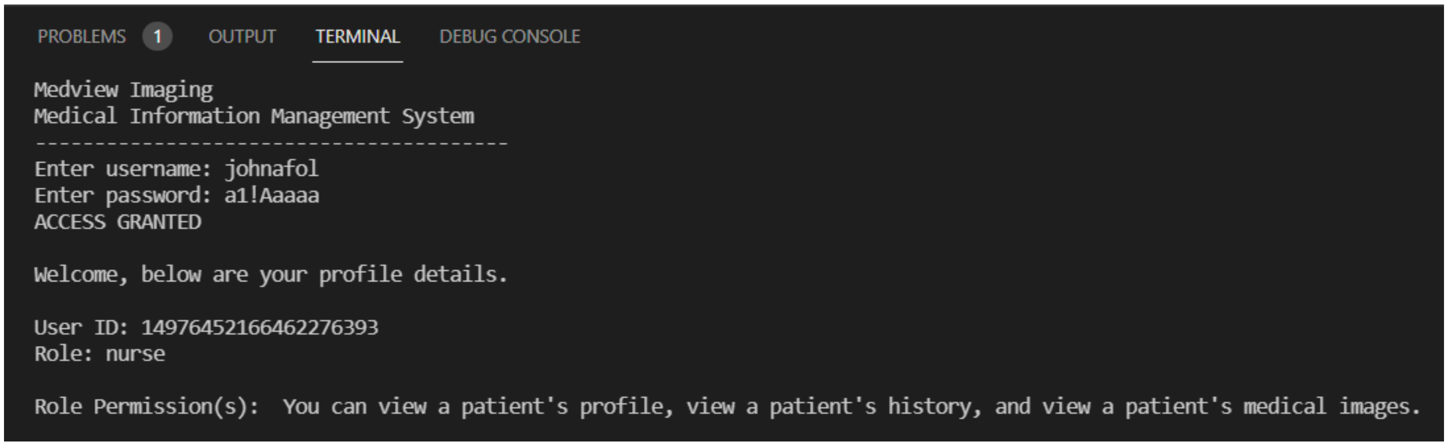
User enters first name

(1c) If password is not the same as password in .txt file, then prompt the user on what to do next

```
Password invalid.
Enter 'y' to try again
----
Enter 'n' to create a new profile
----
Enter 'q' to exit back to the main menu
```

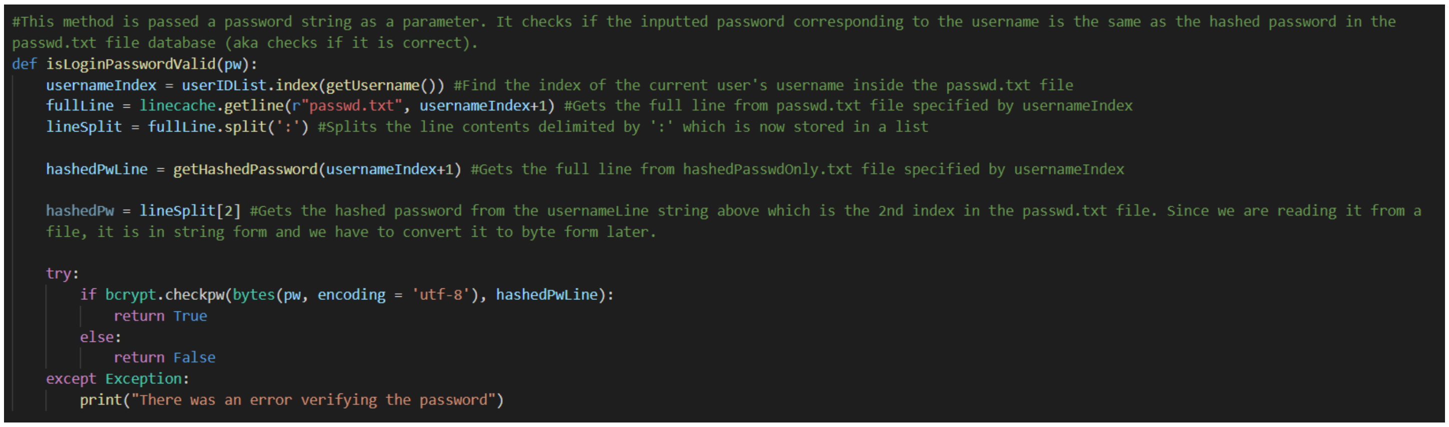
User enters what they want to do next

b)



```
Medview Imaging
Medical Information Management System
----------------------------------------
Enter username: johnafol
Enter password: a1!Aaaaa
ACCESS GRANTED

Welcome, below are your profile details.

User ID: 14976452166462276393
Role: nurse

Role Permission(s):  You can view a patient's profile, view a patient's history, and view a patient's medical images.
```

Figure 11 - Sample Terminal Login Output



```python
#This method is passed a password string as a parameter. It checks if the inputted password corresponding to the username is the same as the hashed password in the passwd.txt file database (aka checks if it is correct).
def isLoginPasswordValid(pw):
    usernameIndex = userIDList.index(getUsername()) #Find the index of the current user's username inside the passwd.txt file
    fullLine = linecache.getline(r"passwd.txt", usernameIndex+1) #Gets the full line from passwd.txt file specified by usernameIndex
    lineSplit = fullLine.split(':') #Splits the line contents delimited by ':' which is now stored in a list

    hashedPwLine = getHashedPassword(usernameIndex+1) #Gets the full line from hashedPasswdOnly.txt file specified by usernameIndex

    hashedPw = lineSplit[2] #Gets the hashed password from the usernameLine string above which is the 2nd index in the passwd.txt file. Since we are reading it from a file, it is in string form and we have to convert it to byte form later.

    try:
        if bcrypt.checkpw(bytes(pw, encoding = 'utf-8'), hashedPwLine):
            return True
        else:
            return False
    except Exception:
        print("There was an error verifying the password")
```

Figure 12 - Password Verification Process

To implement our design, my firm get the current user’s username from the passwd.txt file and then got the index of that username within the file. Using the index we got from our file, we then check the index of hashedPasswdOnly.txt for the user’s hashed password. We then check if the user’s inputted password is the same as our hashed password using bcrypt.checkpw().

c) As seen in Figure 11 above, once a user logs in, they are provided with user info such as their ID, their role, and what permissions their role entails. In the case for user johnafol, his role is as a nurse, his ID is 14976452166462276393, and his role permissions are: viewing a patient's profile, view a patient's history, and view a patient's medical images.

Notes/Assumptions: The PDF never stated that a user should/could attempt to perform any actions within their authority (e.g.: view a patient’s history). Therefore, for simplicity’s sake, my firm only shows a user their permission and terminate the program afterwards.

d) I tested the user login and access control mechanism by logging into a user account (as seen in Figure 11). I also confirmed that the user ID, role, password, and role permissions were the same as that in the .txt file. To further test this, I created additional users, stored the string password and other necessary info in a notebook, and I logged in with that same info and verified that the credentials were correct and matched that of the .txt file.

Problem 5:

Dear MedView Imaging,

My firm designed a user profile platform for you. The platform is meant for your employees to log into their profile or create an existing profile if they don't have one already. The users on this platform will be given roles, depending on the role a user has, they will have certain permissions/restrictions. For example, an administrator in your company can only access the system during business hours from 9:00AM to 5:00PM. When a user creates a profile, the system will collect their info (e.g.: username, password) and save that info into two .txt files. The first .txt file is called passwd.txt and the second is called hashedPasswdOnly.txt. User info: user ID, username and user role will be stored in passwd.txt and the user's hashed password will be stored in hashedPasswdOnly.txt. When a user enters their password upon creating their profile, the system checks the following to ensure that the password is strong:

• Passwords must be least 8-12 characters in length

• Password must include at least:

– one upper-case letter; – one lower-case letter; – one numerical digit, and – one special character from the set: {!, @, #, $, %, ?, ∗}

• Passwords found on a list of common weak passwords (e.g., Password1, Qwerty123, or Qaz123wsx) must be prohibited

If the user's password meets the above requirements, it is then hashed and stored in a hashedPasswdOnly.txt file.

When the system is hashing a password into hashedPasswdOnly.txt, the system uses Bcrypt algorithm to do so. It automatically generates a salt which it attaches to the user's password string, resulting in a hashed password. When a user wants to log into their profile, the system will get the contents of passwd.txt and hashedPasswdOnly.txt, the system will then parse the data within the file and check to see if they are valid. For example, when a user enters their username, the program checks the passwd.txt file to confirm if their username is in the database (passwd.txt file) or not. If so, the program will then ask the user for their password. When a user enters their password, the program will then check hashedPasswdOnly.txt for the hashed password corresponding to the username which the user inputted previously. After getting the hashed password, the system then uses Bcrypt to check if the password the user inputted is the same as the hashed password stored in hashedPasswdOnly.txt. The system contains error handling for many scenarios such as invalid/unexpected inputs when inputting password, username, etc. If a user is able to log into their profile, the system shows them their User ID, their role, and their permissions pertaining to the role they have.

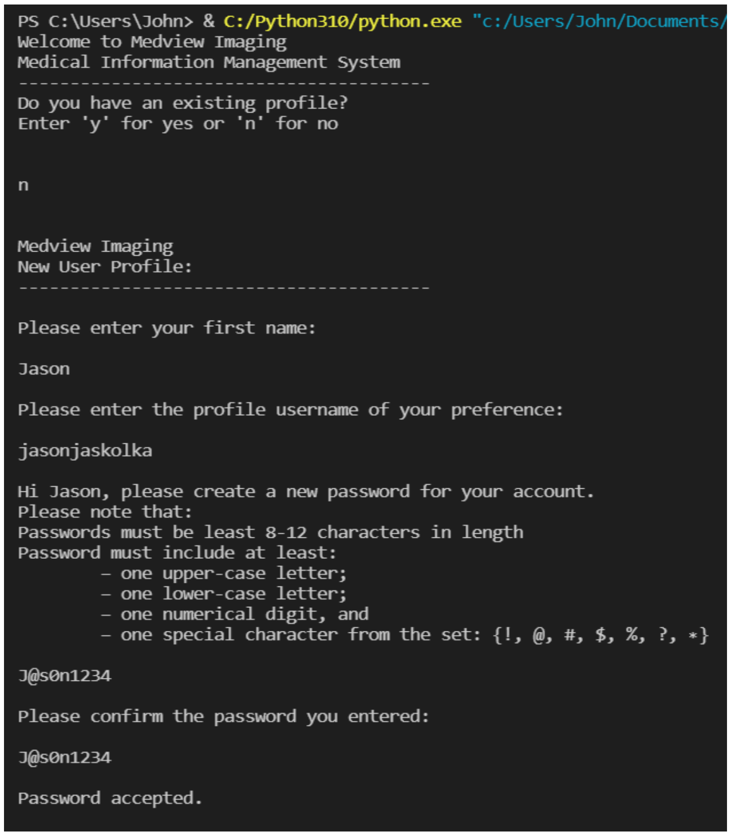
Let's walk through a sample use case: Say we have a new employee who is an administrator, we need to enrol them into our database. Let's say this new user has the following attributes:

Name: Jason   Username: jasonjaskolka   Role: Administrator   Password: J@s0n1234

Let's pretend I'm Jason. I'll create a new profile so you can see what the user creation process is like:

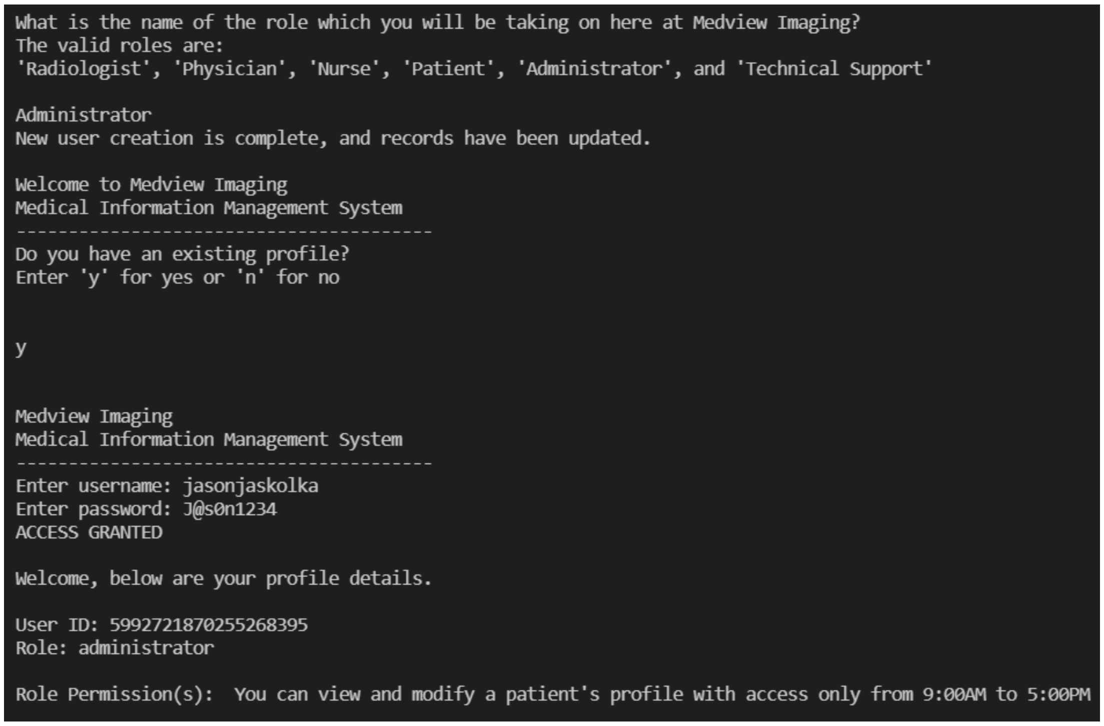
Upon running it, the program will ask if we have an existing profile or not, since we don't, we'll enter 'n' which will prompt us to create a profile.

As you can see in Figures 13 and 14, the program successfully created a new profile for Jason. And if you look at Figure 15, we have confirmed that Jason was successfully added to the database.



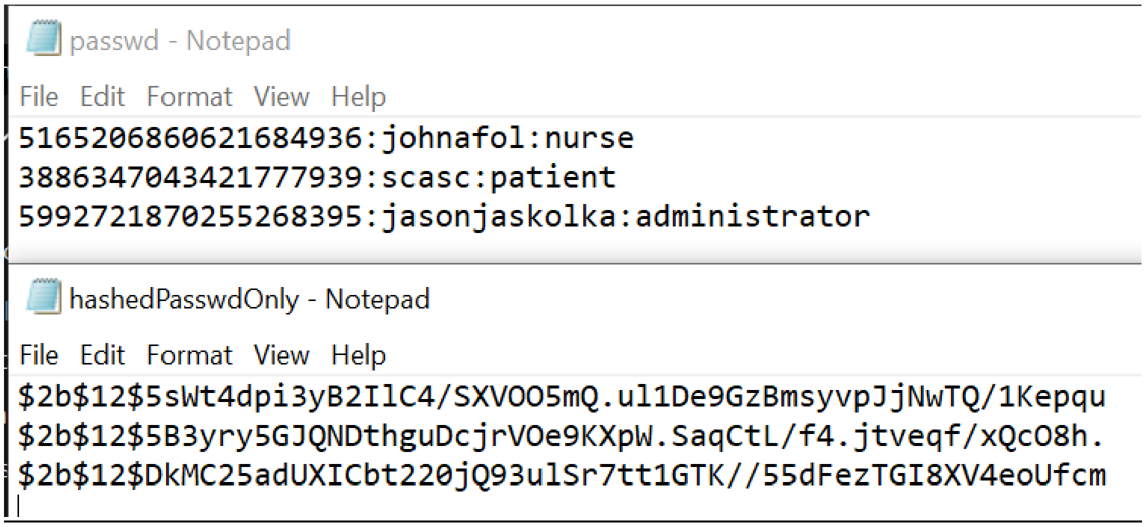
```
PS C:\Users\John> & C:/Python310/python.exe "c:/Users/John/Documents/
Welcome to Medview Imaging
Medical Information Management System
----------------------------------------
Do you have an existing profile?
Enter 'y' for yes or 'n' for no


n


Medview Imaging
New User Profile:
----------------------------------------

Please enter your first name:

Jason

Please enter the profile username of your preference:

jasonjaskolka

Hi Jason, please create a new password for your account.
Please note that:
Passwords must be least 8-12 characters in length
Password must include at least:
        - one upper-case letter;
        - one lower-case letter;
        - one numerical digit, and
        - one special character from the set: {!, @, #, $, %, ?, *}

J@s0n1234

Please confirm the password you entered:

J@s0n1234

Password accepted.
```

Figure 13 - Creating Profile for Jason (1)



```
What is the name of the role which you will be taking on here at Medview Imaging?
The valid roles are:
'Radiologist', 'Physician', 'Nurse', 'Patient', 'Administrator', and 'Technical Support'

Administrator
New user creation is complete, and records have been updated.

Welcome to Medview Imaging
Medical Information Management System
----------------------------------------
Do you have an existing profile?
Enter 'y' for yes or 'n' for no


y


Medview Imaging
Medical Information Management System
----------------------------------------
Enter username: jasonjaskolka
Enter password: J@s0n1234
ACCESS GRANTED

Welcome, below are your profile details.

User ID: 5992721870255268395
Role: administrator

Role Permission(s):  You can view and modify a patient's profile with access only from 9:00AM to 5:00PM
```

Figure 14 - Creating Profile for Jason (2)



passwd - Notepad

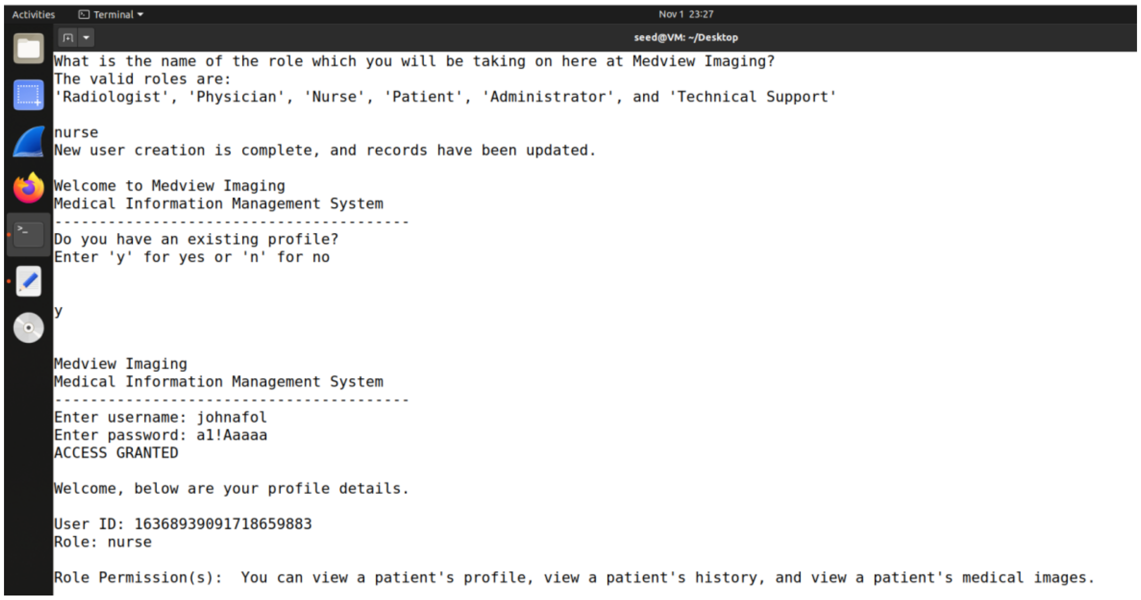
```
5165206860621684936:johnafol:nurse
3886347043421777939:scasc:patient
5992721870255268395:jasonjaskolka:administrator
```

hashedPasswdOnly - Notepad

```
$2b$12$5sWt4dpi3yB2IlC4/SXVOO5mQ.ul1De9GzBmsyvpJjNwTQ/1Kepqu
$2b$12$5B3yry5GJQNDthguDcjrVOe9KXpW.SaqCtL/f4.jtveqf/xQcO8h.
$2b$12$DkMC25adUXICbt220jQ93ulSr7tt1GTK//55dFezTGI8XV4eoUfcm
```

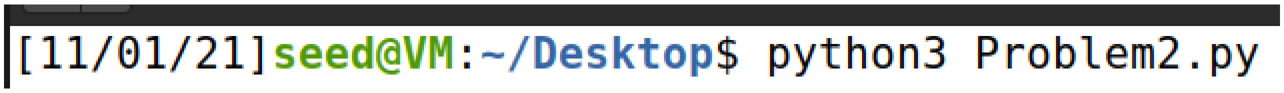
Figure 15 - Confirmed Jason was successfully added to .txt file databases

To run this program, you can use any IDE which supports Python 3 or up. My firm tested it using MS Visual Studio Code on Windows 10 and on Linux using the terminal (Figure 16). If you wish to run the program using Linux's terminal, make sure you type 'python3' proceeding with the file name separated by a space (see Figure 17).



```
What is the name of the role which you will be taking on here at Medview Imaging?
The valid roles are:
'Radiologist', 'Physician', 'Nurse', 'Patient', 'Administrator', and 'Technical Support'

nurse
New user creation is complete, and records have been updated.

Welcome to Medview Imaging
Medical Information Management System
----------------------------------------
Do you have an existing profile?
Enter 'y' for yes or 'n' for no


y


Medview Imaging
Medical Information Management System
----------------------------------------
Enter username: johnafol
Enter password: a1!Aaaaa
ACCESS GRANTED

Welcome, below are your profile details.

User ID: 16368939091718659883
Role: nurse

Role Permission(s):  You can view a patient's profile, view a patient's history, and view a patient's medical images.
```

Figure 16 - Program functioning properly on Linux terminal



```
[11/01/21]seed@VM:~/Desktop$ python3 Problem2.py
```

Figure 17 - How to execute program on Linux terminal

Now you're all set! Thank you, MedView Imaging, for choosing my firm to help you implement your Login system. If you have any trouble, please do not hesitate to contact me.

Sincerely,

John

<u>Final Assumptions/Notes/Thoughts/Issues:</u>

Firstly, the assignment PDF had some redundant requirements. For example, on page 4 of the PDF, it states ‘Passwords matching the format of calendar dates, license plate numbers, telephone numbers, or other common numbers must be prohibited’

I was working on fulfilling those requirements when I realized that if the password must contain at least one number, one lower case and upper-case letter, a special character in the first place, then passwords with license plate formats, date format, and telephone number format would automatically be rejected.

Secondly, I did not specify a path so the program will create .txt files in default home directory, on Windows it’s in the default user directory and on Linux, it’s on the same path which the program is ran from.

Thirdly, I encountered a stressful day attempting to parse the hashed password from my passwd.txt file and checking it with the string which the user inputted. This was because I was reading the username, userRole, userID as string and when I read the hashed password as a string, it could not work because hashedPasswords are saved as bytes using Bcrypt. So, when I read the hashedPassword, I was reading it as a string which is incorrect because it is supposed to be read in bytes. Then I tried reading the file in binary mode, but I faced trouble parsing the file and delimiting ‘:’ in the file. I was stuck on this for over a day with no one to turn to for help and in an effort not to waste time, I created a second password file hashedPasswdOnly.txt which would contain only hashed password, so I didn’t have to parse any colons. Thankfully, this fixed the issue I was having. Please let me know if there is an easy way to do this, thank you.

Lastly, these problems all build on top of each other so I complied everything into one file (Problem2.py) which contains the solution to all code requirements.

Sources/Resources Used:

https://stackoverflow.com/questions/19057939/add-user-input-to-list

https://www.youtube.com/watch?v=hNa05wr0DSA

https://stackoverflow.com/questions/10012534/how-to-generate-a-big-random-number-in-python

https://www.geeksforgeeks.org/python-test-if-string-contains-any-uppercase-character/

https://www.geeksforgeeks.org/python-program-check-string-contains-special-character/

https://stackoverflow.com/questions/19511440/add-b-prefix-to-python-variable

https://pythonise.com/categories/python/python-password-hashing-bcrypt

https://security.stackexchange.com/questions/39849/does-bcrypt-have-a-maximum-password-length

https://www.w3schools.com/python/python_file_write.asp

https://stackoverflow.com/questions/17056526/what-is-the-easiest-way-to-read-the-text-file-delimited-by-tab-in-python

https://pynative.com/python-read-specific-lines-from-a-file/

https://stackabuse.com/python-check-index-of-an-item-in-a-list/

https://www.devdungeon.com/content/working-binary-data-python

https://pypi.org/project/bcrypt/

https://github.com/danielmiessler/SecLists/blob/master/Passwords/2020-200_most_used_passwords.txt

https://stackoverflow.com/questions/25341945/check-if-string-has-date-any-format