CS3053 - Computer Security

Continuous assessment - Protection of information based on sensitivity and privilege levels

Name : Weerasekara W.M.T.B.

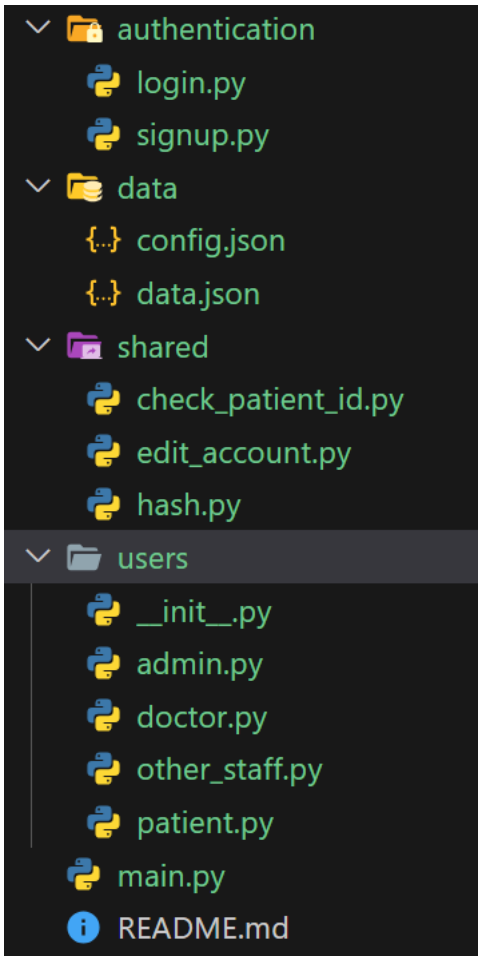
Index : 200698X

1. **The program code to carry out the above steps**

GitHub : https://github.com/ThejanB/Medical_Processing_System-Security

The medical data processing program was programmed using Python language

File Structure us as following:



Hashing encoding and decoding functions :

```python
import hashlib
import base64

### hash_password function to hash password      You, 5 minutes ago
def hash_password(user_password):
    hashed_password = hashlib.md5()
    hashed_password.update(user_password.encode("utf-8"))
    return hashed_password.hexdigest()

### encode function to encode string to base64
def encode(encode_text):
    base64_bytes = base64.b64encode(encode_text.encode('ascii'))
    base64_message = base64_bytes.decode('ascii')
    return base64_message


### decode function to decode base64 encoded string
def decode(decode_text):
    message_bytes = base64.b64decode(decode_text.encode('ascii'))
    message = message_bytes.decode('ascii')
    return message
```

User types are

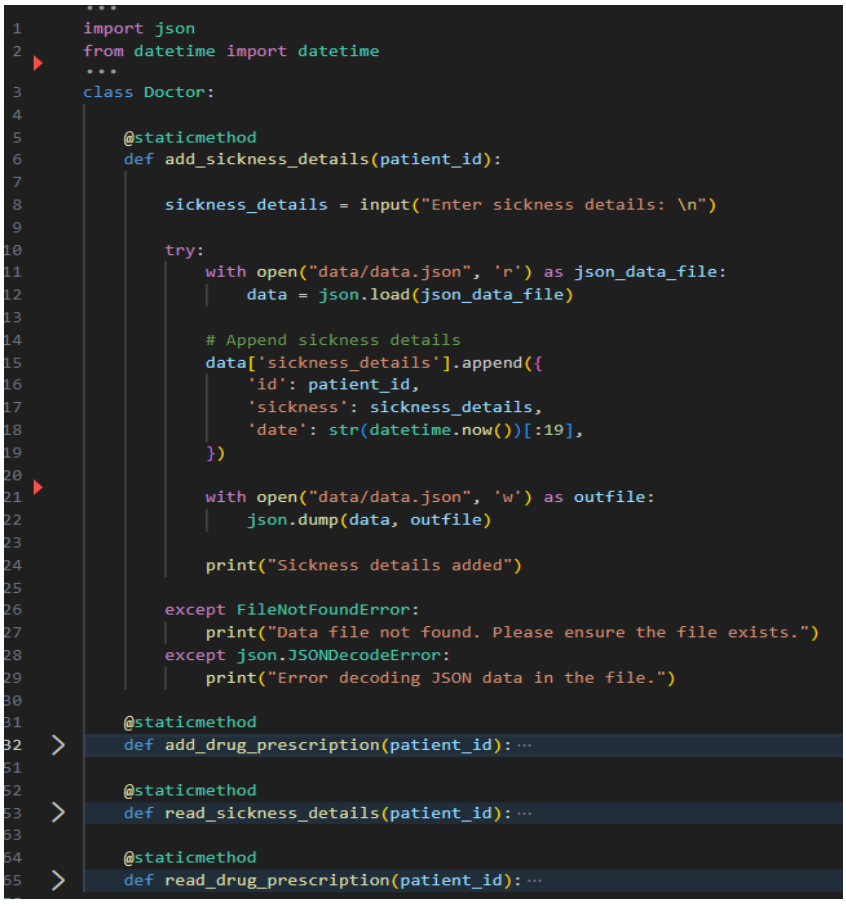
1. Admin
2. Doctor
3. Other Staff
4. Patient

Created a Class for each to handle its own tasks

Admin Class

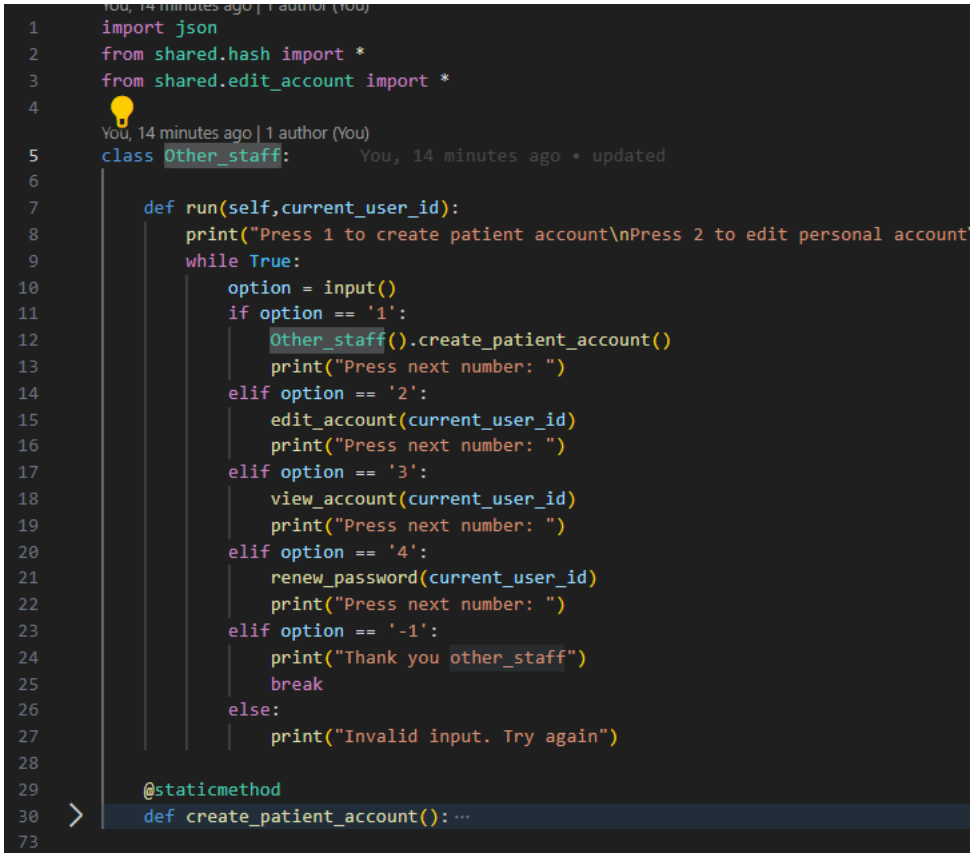
```
You, 11 minutes ago | 1 author (You)
from shared.hash import *
import json

You, 11 minutes ago | 1 author (You)
class Admin:
    def run(self):

        while True:
            role_number = input("Press 1 to edit doctor code\nPres

            if role_number == '1':
                self.change_doctor_code()

            elif role_number == '2':
                self.change_other_staff_code()

            elif role_number == '3':
                self.create_admin()

            elif role_number == '-1':
                print("Admin logout!\n")
                break
            else:
                print("Invalid input")
        You, 11 minutes ago • updated
    @staticmethod
    def create_admin(): ···

    @staticmethod
    def change_doctor_code(): ···

    @staticmethod
    def change_other_staff_code(): ···
```

Doctor Class



```
import json
from datetime import datetime
class Doctor:

    @staticmethod
    def add_sickness_details(patient_id):

        sickness_details = input("Enter sickness details: \n")

        try:
            with open("data/data.json", 'r') as json_data_file:
                data = json.load(json_data_file)

            # Append sickness details
            data['sickness_details'].append({
                'id': patient_id,
                'sickness': sickness_details,
                'date': str(datetime.now())[:19],
            })

            with open("data/data.json", 'w') as outfile:
                json.dump(data, outfile)

            print("Sickness details added")

        except FileNotFoundError:
            print("Data file not found. Please ensure the file exists.")
        except json.JSONDecodeError:
            print("Error decoding JSON data in the file.")

    @staticmethod
    def add_drug_prescription(patient_id): ···

    @staticmethod
    def read_sickness_details(patient_id): ···

    @staticmethod
    def read_drug_prescription(patient_id): ···
```

Other Staff Class



```python
import json
from shared.hash import *
from shared.edit_account import *

class Other_staff:

    def run(self,current_user_id):
        print("Press 1 to create patient account\nPress 2 to edit personal account
        while True:
            option = input()
            if option == '1':
                Other_staff().create_patient_account()
                print("Press next number: ")
            elif option == '2':
                edit_account(current_user_id)
                print("Press next number: ")
            elif option == '3':
                view_account(current_user_id)
                print("Press next number: ")
            elif option == '4':
                renew_password(current_user_id)
                print("Press next number: ")
            elif option == '-1':
                print("Thank you other_staff")
                break
            else:
                print("Invalid input. Try again")

    @staticmethod
    def create_patient_account():···
```

Patient Class

```python
import json


class Patient:

    @staticmethod
    def read_sickness_details(patient_id):

        # read sickness details from data file
        with open("data/data.json", 'r') as json_data_file:
            data = json.load(json_data_file)
            details = data['sickness_details']

            for detail in details:
                if detail['id'] == patient_id:
                    print(detail['date'] , '-' , detail['sickness'])

    @staticmethod
    def read_drug_prescription(patient_id):

        # read drug prescription from data file
        with open("data/data.json", 'r') as json_data_file:
            data = json.load(json_data_file)
            details = data['drug_presc']

            for detail in details:
                if detail['id'] == patient_id:
                    print(detail['date'] , '-' , detail['drug_presc'])

```

Sign in functions

```python
import json
from shared.hash import *

class SignUp:
    roles = {'1': "doctor", '2': "other_staff"}

    def create_new_user(self):

        name = input('Username:')
        password = input('Enter a password including an upper-case letter, lower-case letter, a digit and length not less than 5: \n')
        while True:
            if not (any(x.isupper() for x in password) and
                    any(x.islower() for x in password) and
                    any(x.isdigit() for x in password) and
                    len(password) >= 5):
                password = input("Password is too weak. Please re-enter a new password: ")
            else:
                break
        safe_password = hash_password(password)

        # read config file for doctor and other_staff codes
        with open("data/data/config.json", 'r') as json_data_file:
            data = json.load(json_data_file)
            doc_code = data['codes'].get('doctor')
            rec_code = data['codes'].get('other_staff')

        # hotel staff accounts can only be created by using the relevant code for security
        while True:

            role_number = input('Press 1 for "doctor account" or 2 for "other_staff account": ')

            if role_number == '1':
                code = hash_password(input("Please enter the doctor code to create account: "))
                if code == doc_code:
                    user_type = self.roles.get('1')
                    privilege_level = '2'
                    break

            elif role_number == '2':
                code = hash_password(input("Please enter the other_staff code to create account: "))
                if code == rec_code:
                    user_type = self.roles.get('2')
                    privilege_level = '3'
                    break
                else:
                    print("Invalid code")
            else:
                print("Invalid input. Try again")

        # read and write user to config file
        with open("data/config.json", 'r') as json_data_file:
            data = json.load(json_data_file)
            user_id = len(data['users'])+1
            data['users'].append({
                'id': user_id,
                'name': name,
                'password': safe_password,
                'user_type': user_type,
                'privilege_level': privilege_level
            })
        with open("data/config.json", 'w') as outfile:
            json.dump(data, outfile)

        print("Account created successfully")

        # fill account details
        print("Fill in personal account details")
        acc_name = input('Your name: ')
        age = input('Age: ')
        nic = input('NIC number: ')
        tel = input('Telephone number: ')

        # write account details to data file
        with open("data/data.json", 'r') as json_data_file:
            data = json.load(json_data_file)
            data['personal_details'].append({
```

```
76          # write account details to data file
77          with open("data/data.json", 'r') as json_data_file:
78              data = json.load(json_data_file)
79              data['personal_details'].append({
80                  'id': user_id,
81                  'name': acc_name,
82                  'age': age,
83                  'nic': encode(nic),
84                  'tel': encode(tel)
85              })
86          with open("data/data.json", 'w') as outfile:
87              json.dump(data, outfile)
88
89          print("Account completed")
90
```

Login function

```
You, 15 minutes ago | 1 author (You)
1   import json
2   from shared.hash import hash_password
3
You, 15 minutes ago | 1 author (You)
4   class Login:
5
6       @staticmethod
7       def auth_user():
8           result = 'Login failed'
9
10          name = input("Enter username: ")
11          hashed_password = hash_password(input("Enter password: "))
12
13          with open("data/config.json", 'r') as json_data_file:
14              data = json.load(json_data_file)      You, 15 minutes ago • updated
15              for user in data['users']:
16                  if name == user['name'] and hashed_password == user['password']:
17                      current_user = user
18                      result = "Login successful"
19          print(result)
20          if result == "Login successful":
21              return current_user
22          else:
23              return False
24
25
26
```

2. A description of how I decided the access to data records based on sensitivity of data

The different data records have different levels of sensitivity and reading and writing rights should only be given to certain user roles.

I took four types of user roles with different privilege levels

1. **Admin** - privilege level 1
    - set up secret codes for Doctor and Other staff accounts, this will be saved as hashed codes in the config file.
    - Add new admin functionality
2. **Doctor** - privilege level 2
    - Can create patient accounts. Before being allowed to create patient accounts, the doctor user will be prompted to enter the doctor code.
    - Doctor can add and view patients details and illness data of patients
    - Can view and edit their personal details.
3. **Other staff** - privilege level 3
    - Can create patient accounts. Before being allowed to create patient accounts, the other staff user will be prompted to enter the other_staff code.
    - Other staff can view patients details and illness data of patients. They cannot add patients details or illness data.
    - Can view and edit their personal details.
4. **Patient** - privilege level 4
    - Can only read their own sickness details
    - Can view and edit their personal details.

Security Measures

This command line, role-based access control system ensures that users only have access to data and operations that are appropriate for their role and privilege level.

All passwords are hashed using MD5 before being written to the config file, enhancing the security of stored passwords.

Sensitive data, such as NIC numbers and telephone numbers, are encoded in base64 before being written to the data file, providing an additional layer of data security.

3. Configuration and data files

Config file –> save user authentication data, doctor_code and other_staff_code

Data File -> Save patients data as lists

--- All the Codes are available in above GitHub Link ---