Keys	Values
Name	Sanjay Nithin S
Reg No.	20BIT0150
Title	Adding security measures to Hospital Management System

New

Webapp description

1. Flask/Django App.
2. It will have a login/signup page and hashing is used for password authentication.
3. It'll ask to upload official documents like aadhar card for verification which will be encrypted and stored.
4. While accessing it, it'll be decrypted and displayed to the user.
5. MongoDb is used for the data storage.
6. Encryption and decryption will be done in the server and not in the Database.
7.

8. **generating rsa key**

```
from Crypto.PublicKey import RSA
secret_code = "Unguessable"
key = RSA.generate(2048)
encrypted_key = key.export_key(passphrase=secret_code, pkcs=8, protection="scryptAndAES128-CBC")
file_out = open("rsa_key.bin", "wb")
file_out.write(encrypted_key)
file_out.close()
print(key.publickey().export_key())
```

9. **read back in private key**

```
from Crypto.PublicKey import RSA
secret_code = "Unguessable"
encoded_key = open("rsa_key.bin", "rb").read()
key = RSA.import_key(encoded_key, passphrase=secret_code)
print(key.publickey().export_key())
```

10. **generating public and private keys**

```
from Crypto.PublicKey import RSA
key = RSA.generate(2048) // generated 2048 bits
```

```
private_key = key.export_key()
file_out = open("private.pem", "wb")
file_out.write(private_key)
file_out.close()
public_key = key.publickey().export_key()
file_out = open("receiver.pem", "wb")
file_out.write(public_key)
file_out.close()
```

11. **Encrypting data**

```
from Crypto.PublicKey import RSA
from Crypto.Random import get_random_bytes
from Crypto.Cipher import AES, PKCS1_OAEP
data = "I met aliens in UFO. Here is the map.".encode("utf-8")
file_out = open("encrypted_data.bin", "wb")
recipient_key = RSA.import_key(open("receiver.pem").read())
session_key = get_random_bytes(16)
# Encrypt the session key with the public RSA key
cipher_rsa = PKCS1_OAEP.new(recipient_key)
enc_session_key = cipher_rsa.encrypt(session_key)
# Encrypt the data with the AES session key
cipher_aes = AES.new(session_key, AES.MODE_EAX)
ciphertext, tag = cipher_aes.encrypt_and_digest(data)
[ file_out.write(x) for x in (enc_session_key, cipher_aes.nonce, tag, ciphertext) ]
file_out.close()
```

12. **Decrypting data**

```
from Crypto.PublicKey import RSA
from Crypto.Cipher import AES, PKCS1_OAEP
file_in = open("encrypted_data.bin", "rb")
private_key = RSA.import_key(open("private.pem").read())
enc_session_key, nonce, tag, ciphertext = \
   [ file_in.read(x) for x in (private_key.size_in_bytes(), 16, 16, -1) ]
# Decrypt the session key with the private RSA key
cipher_rsa = PKCS1_OAEP.new(private_key)
session_key = cipher_rsa.decrypt(enc_session_key)
# Decrypt the data with the AES session key
cipher_aes = AES.new(session_key, AES.MODE_EAX, nonce)
data = cipher_aes.decrypt_and_verify(ciphertext, tag)
print(data.decode("utf-8"))
```

13. **Hashing**

```
from Crypto.Hash import SHA256
hash_object = SHA256.new(data=b'First')
```

```
hash_object.update(data = b'Sanjay Nithin')
print(hash_object.hexdigest())
```

14. **Key Generation for fernet**

```
from cryptography.fernet import Fernet
key = Fernet.generate_key()
# string the key in a file
with open('filekey.key', 'wb') as filekey:
   filekey.write(key)
```

15. **Encrypting a file**

```
from cryptography.fernet import Fernet
with open('filekey.key', 'rb') as filekey:
    key = filekey.read()
fernet = Fernet(key)
with open('sanjay.txt', 'rb') as file:
    original = file.read()
encrypted = fernet.encrypt(original)
with open('sanjay.txt', 'wb') as encrypted_file:
    encrypted_file.write(encrypted)
```

16. **Decrypting a file**

```
from cryptography.fernet import Fernet
with open('filekey.key', 'rb') as filekey:
    key = filekey.read()
fernet = Fernet(key)
with open('sanjay.txt', 'rb') as enc_file:
    encrypted = enc_file.read()
decrypted = fernet.decrypt(encrypted)
with open('sanjay.txt', 'wb') as dec_file:
    dec_file.write(decrypted)
```

17. #todo Stuff to do:
    1. Hashing is done for storing password.
    2. RSA key is used to encrypt and store strings.
    3. Fernet encryption is used to encrypt byte files like images, documents, etc.
18. About website:
    1. Hospital management system.
    2. You'll have a login page.
    3. You'll have to upload documents.
    4. AADHAR card will be uploaded and it'll be encrypted.

5. And if possible, using NLP tools aadhar card number will be extracted and will be used as username and will be encrypted using RSA algorithm.
6. https://github.com/dilippuri/Aadhaar-Card-OCR -> used to extract information from aadhar card.