REPORT

ENCRYPTED FILE TRANSFER

SUBMITTED BY

AGNIM CHAKRABORTY (18BCE2186)

GURNEHMAT KAUR DHINDSA (19BCE0227)

SOMONNOY BANERJEE (18BCE2149)

PREPARED FOR

SOFTWARE ENGINEERING (CSE 3001)

UNDER THE GUIDANCE OF

DR. BHAVANI S.

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

SLOT: G1+TG1

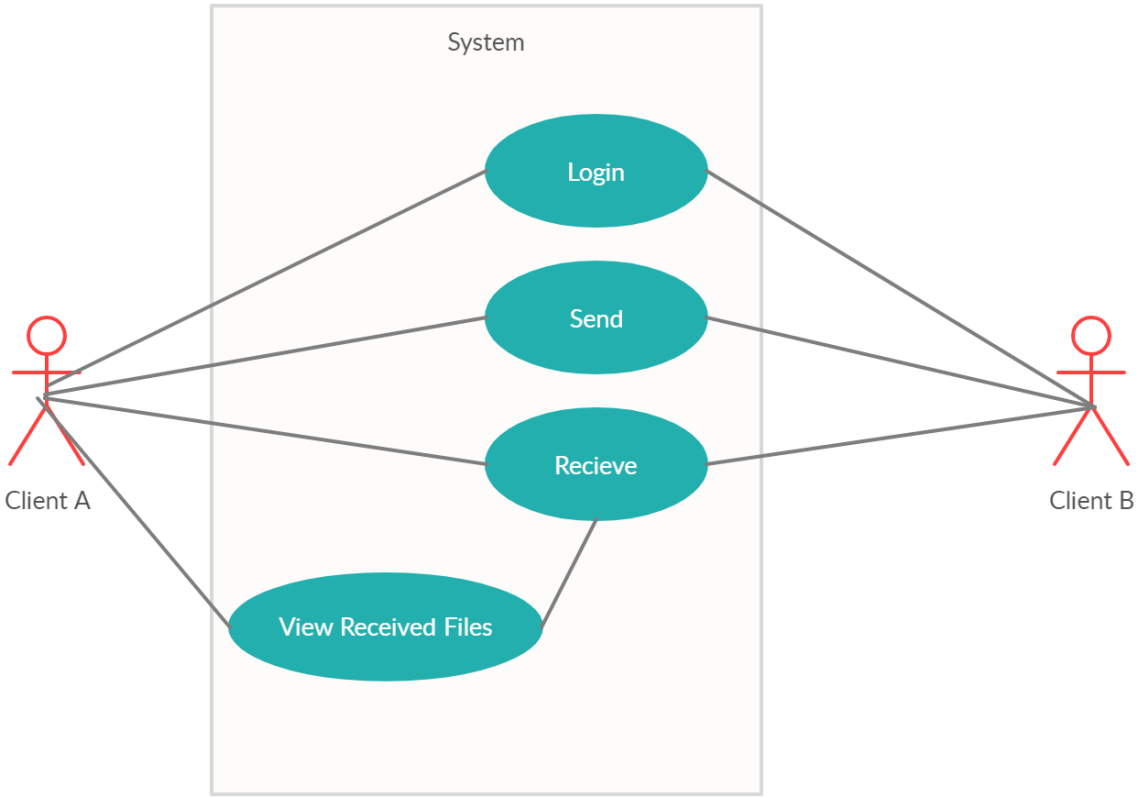
Vellore-632014, Tamil Nadu, India

Problem Statement

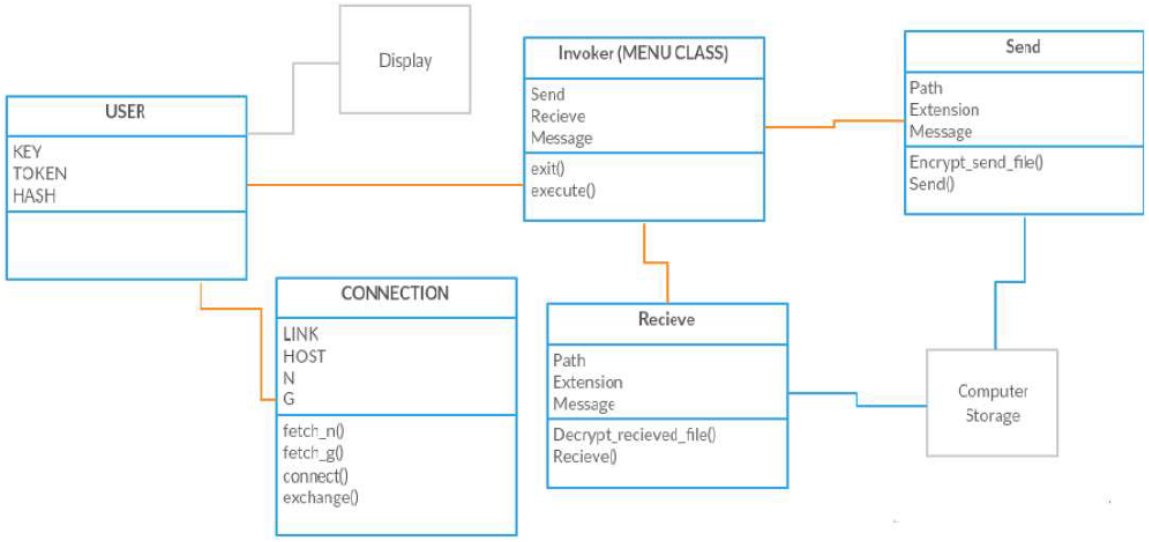
The aim of the project is to create a secure file transfer software for any two (or more) people connected over the internet featuring end to end asymmetric encryption and creating a web server for the same. The software aims to be anonymous thus keeping no information of the data itself and the server aims to allow transfer of unreadable encrypted data. Thus, the data transfer will be dark and only accessible and usable to the persons concerned. One of the major aims of the software is to enforce anonymity of users over the web and perform transfers without leaving a trace over the internet. However general user and password services require databases to hold them to verify login.

UML Diagrams

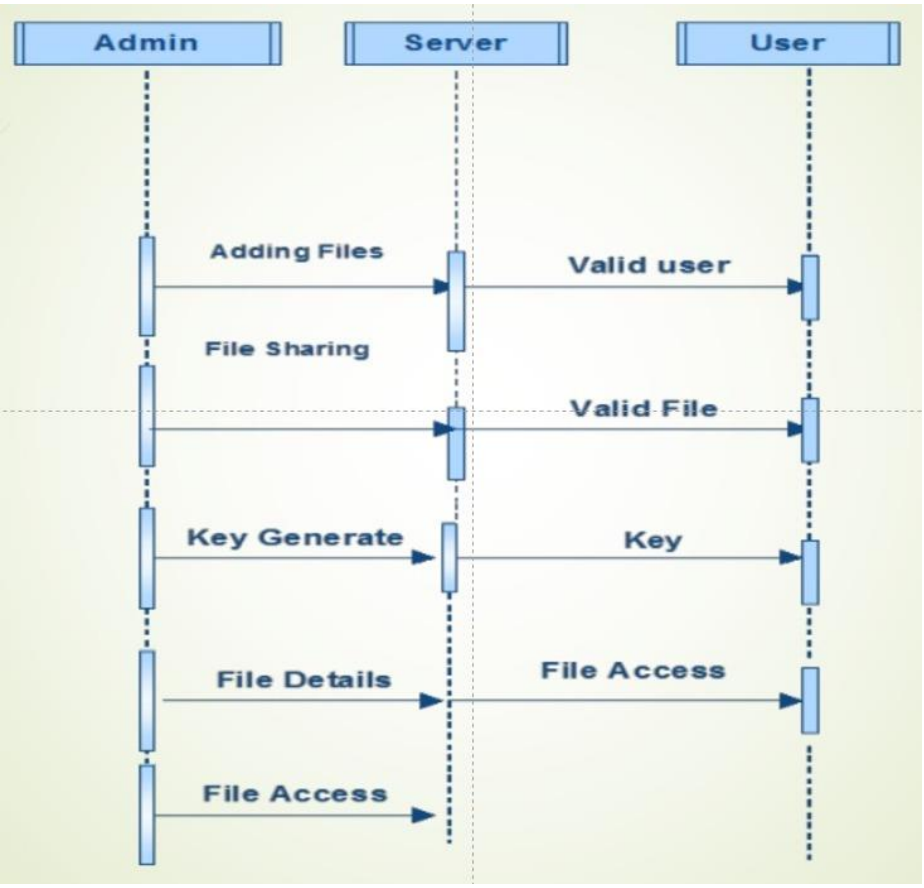
Use Case diagram:



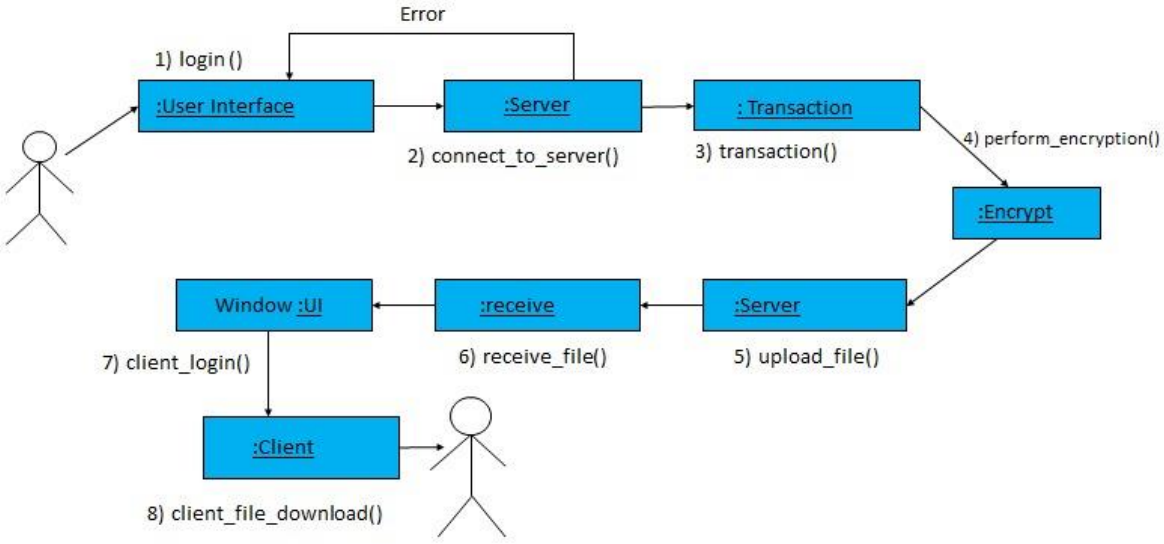
Class diagram:



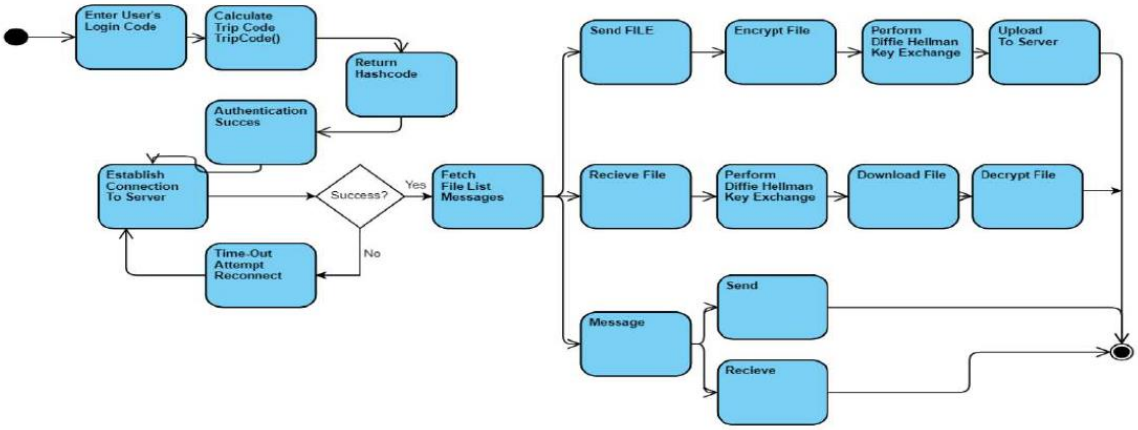
Sequence diagram:



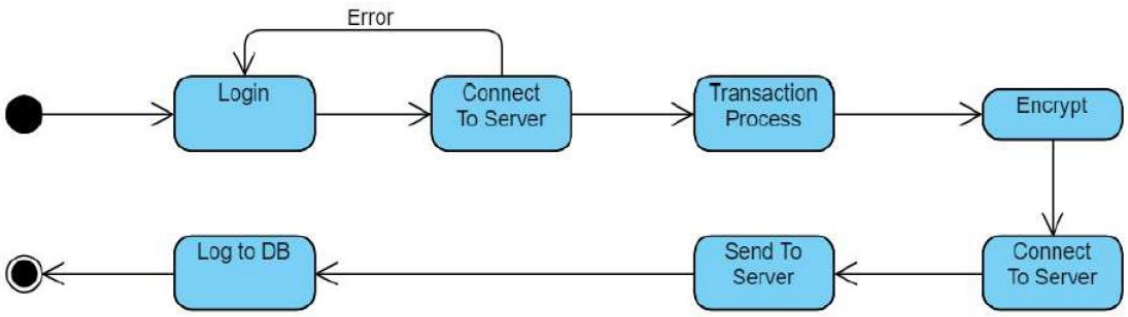
Collaboration diagram:



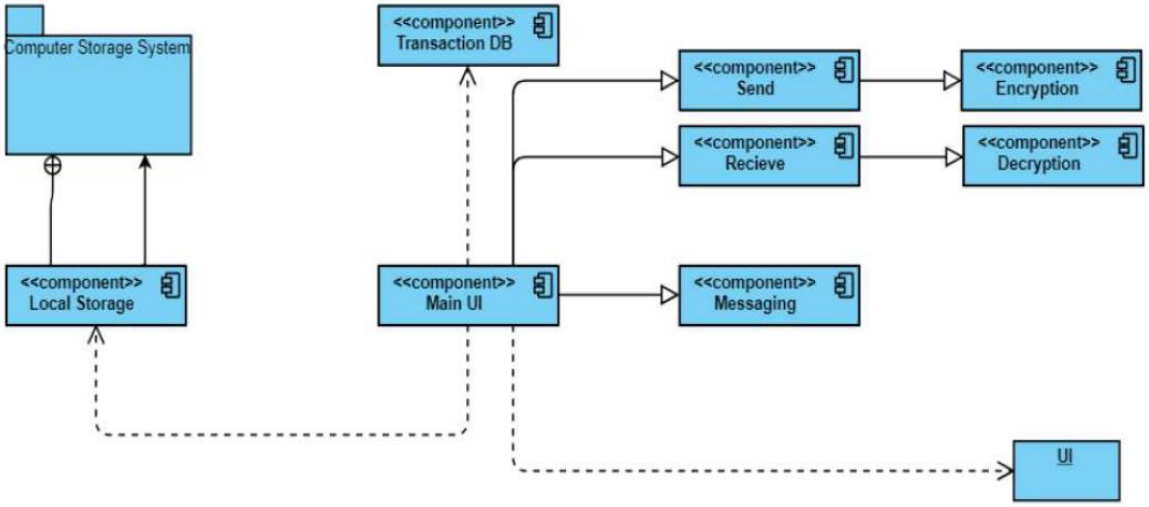
Activity diagram:



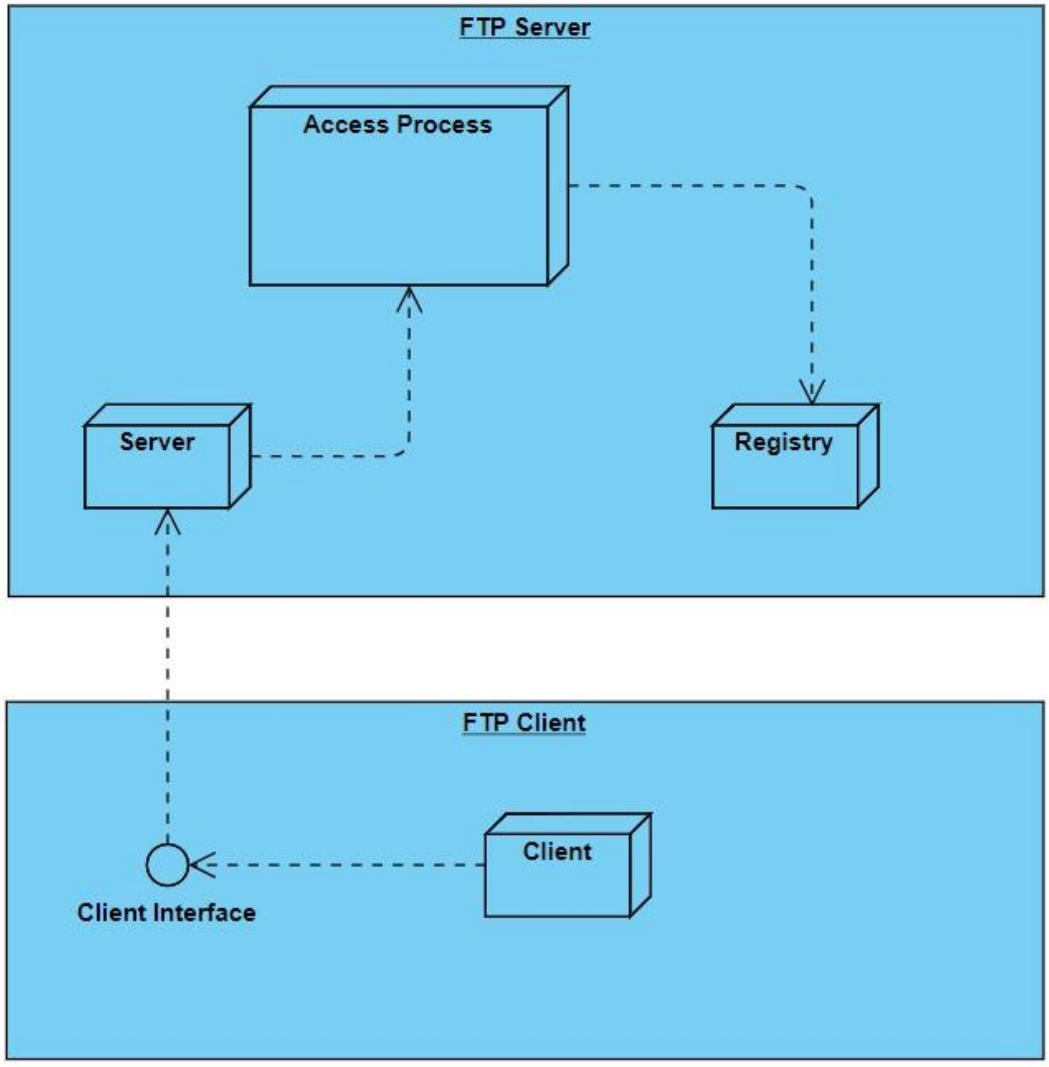
State diagram:



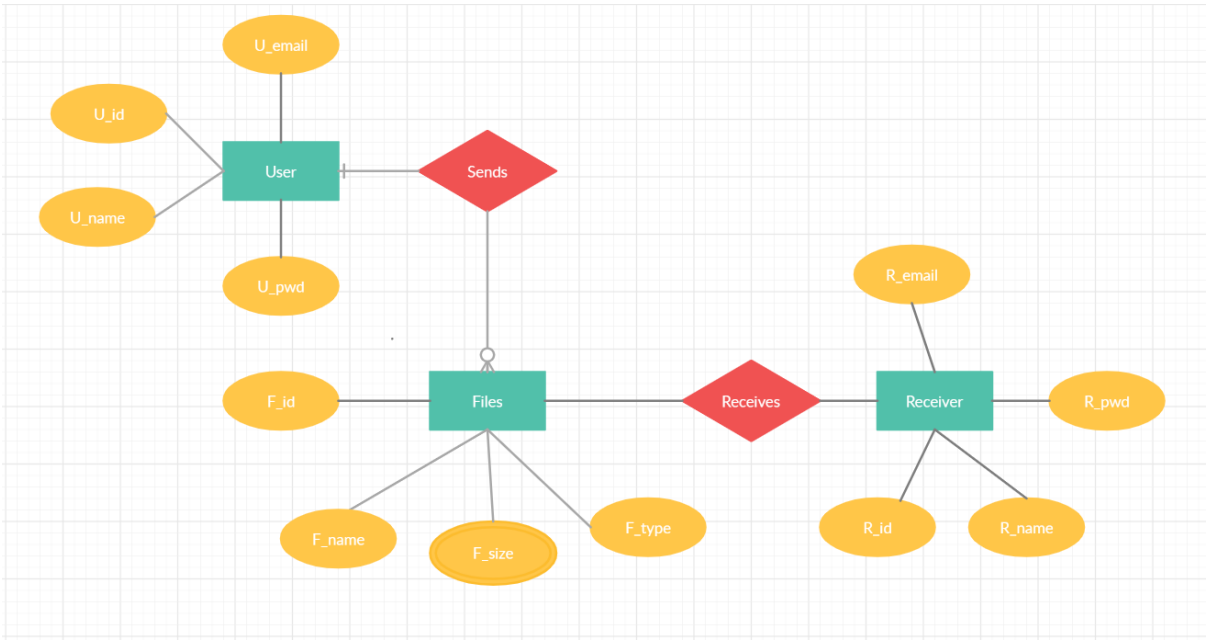
Component diagram:



Deployment diagram:

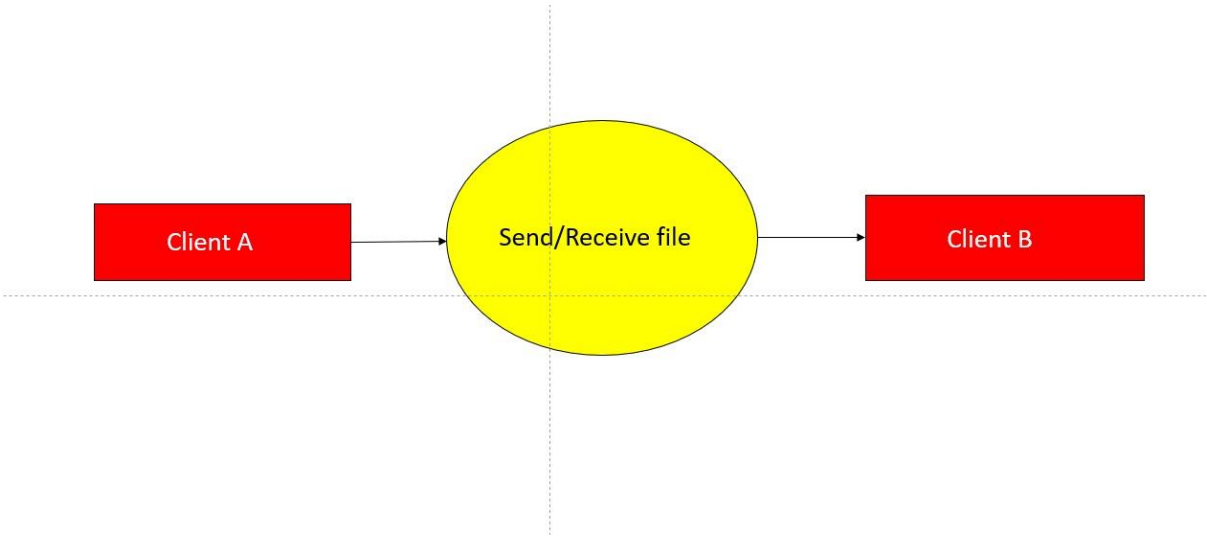


ER DIAGRAM: -

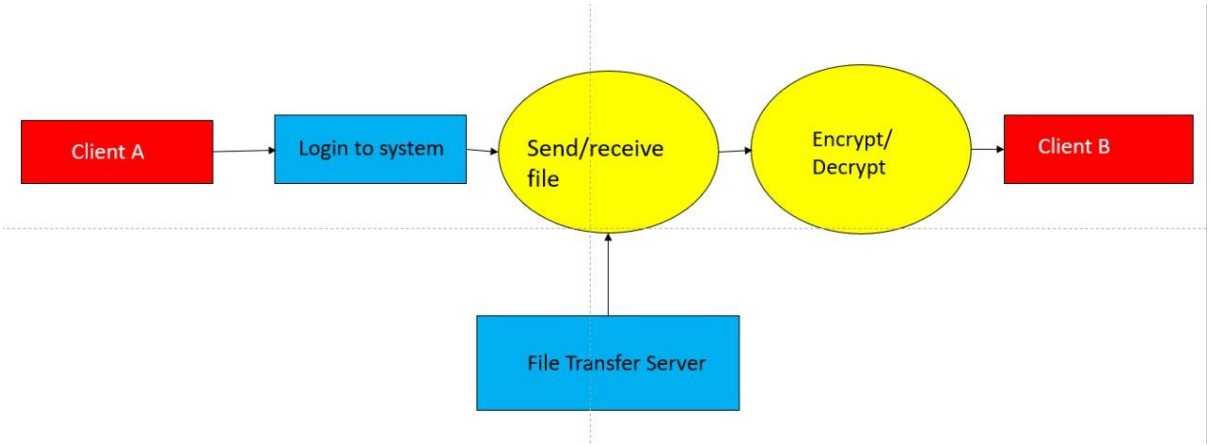


DFD: -

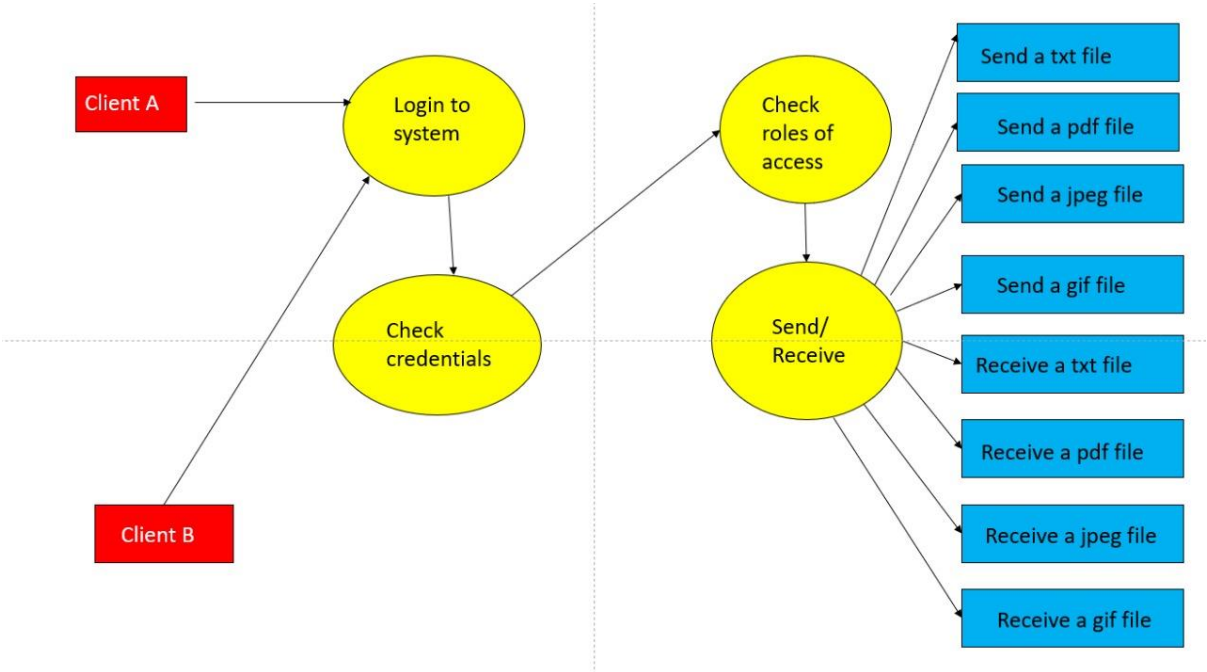
LEVEL 0



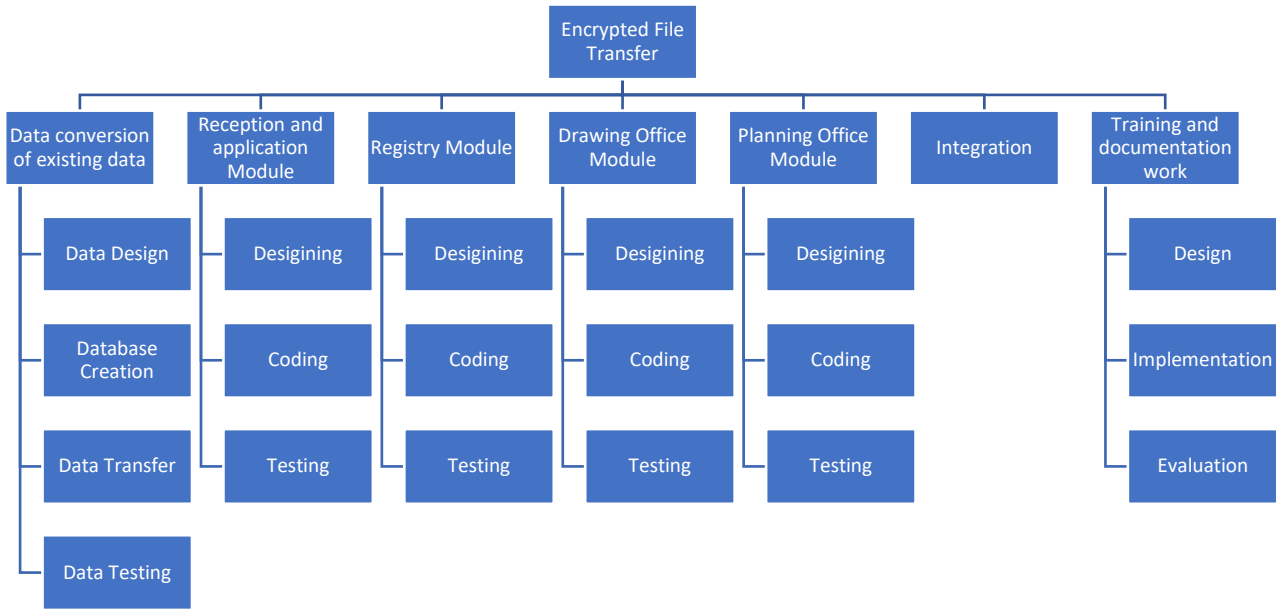
LEVEL 1



LEVEL 2



WORK BREAKDOWN STRUCTURE: -

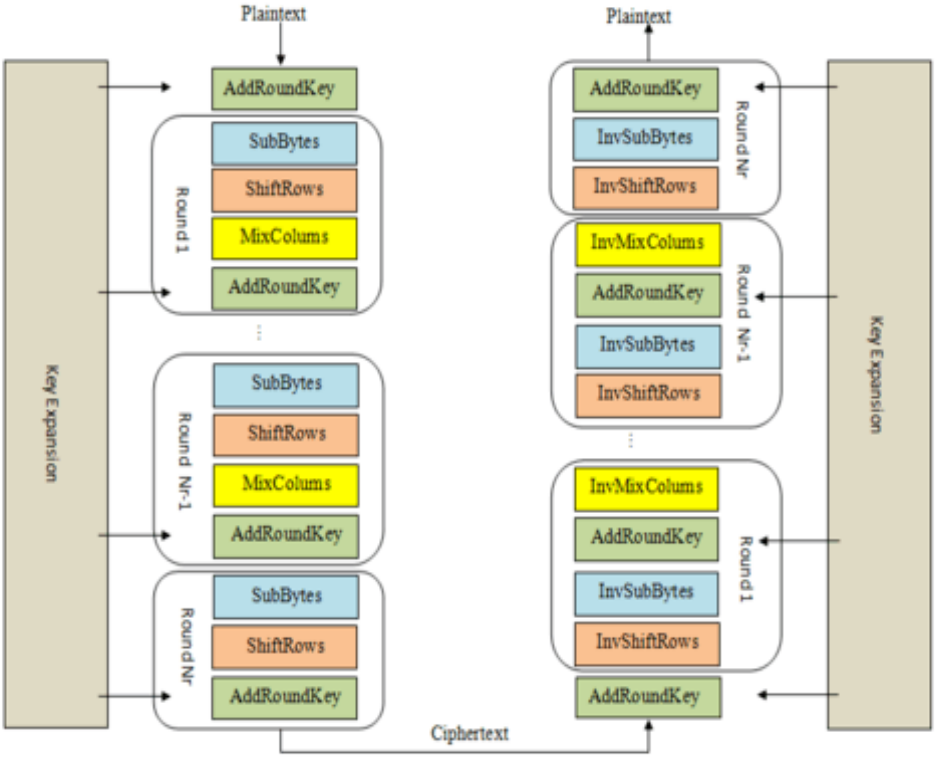


INNOVATION: -

Traditional networking uses normal username and password scheme for granting access. The username is sent to the server using the USER command, and the password is sent using the PASS command. This sequence is unencrypted "on the wire" that is open to man in middle attacks. This point is one of the major focus for our project. The second domain here is Encryption and we will be employing the use of encryption to make data unreadable while enforcing anonymity over the internet.

Older encryption algorithms were prone to cracking by means of mathematical as well as observational methods. The older ciphers were kept secretive to ensure that the data could not be read by unauthorized people. However, the newer encryption algorithms i.e. AES are based on the idea of keeping the cipher function absolutely public however the key a secret. The use of several mathematical functions to create a final unrepeatable encryption function for the message (or file) eliminates the possibility of cracking the encryption by observational methods thus making it solely impossible to crack any encrypted data without access to the keys.

AES (Advanced Encryption Standard) Model

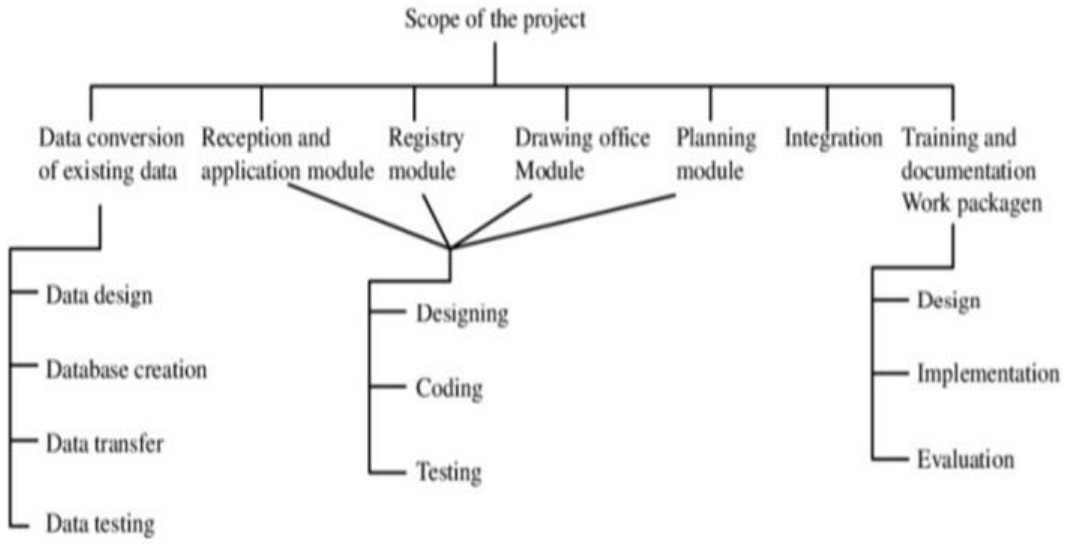


Security concerns have increased in recent times and data privacy has become pivotal. This has been our motivation behind creating this project.

Our proposal is to create a secure file transfer software for any two (or more) people connected over the network. End to end asymmetric encryption is one of the useful methods to achieve this.

The project would create an extremely secure file transfer system however suffers from some bottlenecks. The software requires a higher net speed to transfer larger files. The system also needs a file space equal or more than the file to be encrypted to send it, so as to create a temporary copy of the file in encrypted format and then send it across the net. Both parties also require a medium to fast performing PC or ample time to perform encryption and decryption operations on the file. This in respect to regular file transfer systems i.e. uploading and downloading is comparatively slower for larger file sizes yet similar for small file sizes.

MODULE/ ARCHITECTURE DIAGRAM: -

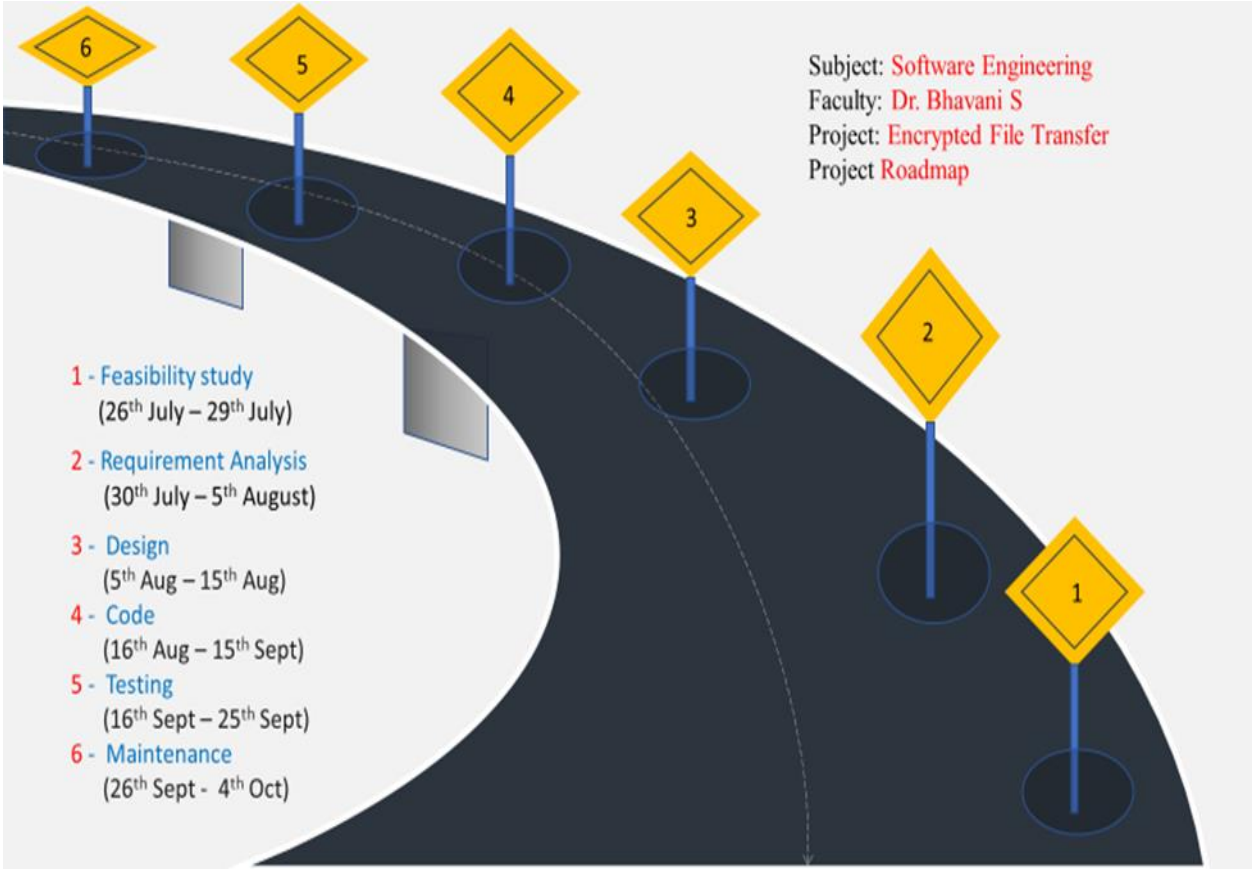


SYSTEM ARCHITECTURE:

A system architecture or systems architecture is the conceptual model that defines the structure, behaviour, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviours of the system. It is a response to the conceptual and practical difficulties of the description and the design of complex systems.

- The system will be divided into client and server application. The server needs an extremely basic GUI and as it doesn't need to be that advanced i.e. it's very restricted as it doesn't maintain transfer logs and connections. The files would not be read by the server nor do they do undergo any processing.
- The client will have a full featured GUI and use hash-based IDs instead of regular ID and password for user anonymity. The encryption shall take place on the client end and shall feature all common audio, video and document formats as well byte level encryption for other forms of data.
- The software aims to be anonymous thus keeping no information of the data itself and the server aims to allow transfer of unreadable encrypted data. Thus, the data transfer is only accessible and usable to the persons concerned.

ROADMAP:



LAYERED PATTERN:

Components within the layered architecture pattern are organized into horizontal layers, each layer performing a specific role within the application (e.g., presentation logic or business logic). Each layer in the architecture forms an abstraction around the work that needs to be done to satisfy a particular request.

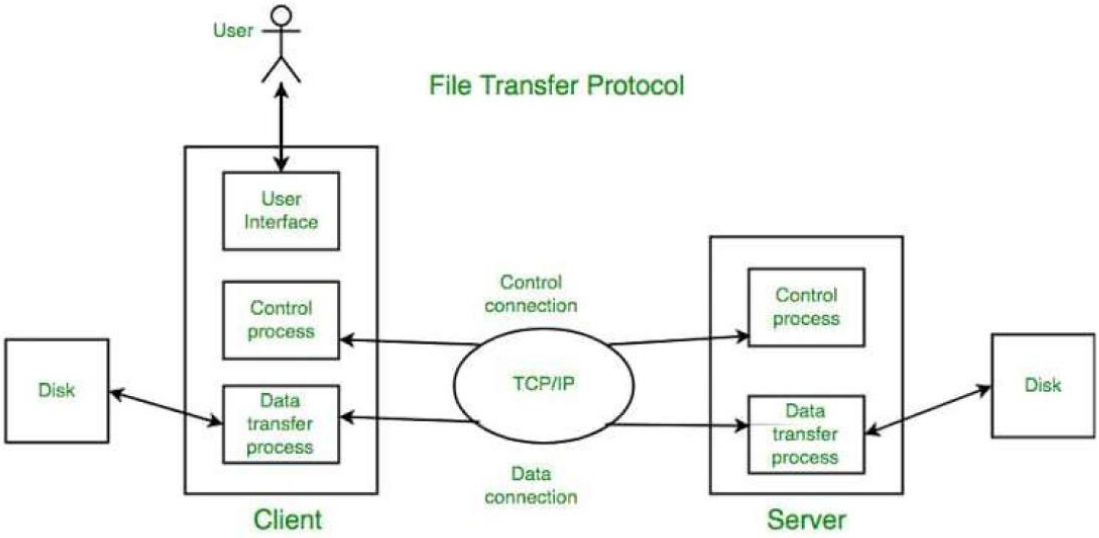
Divided into 4 parts:

- Presentation layer (also known as UI layer)
- Application layer (also known as service layer)
- Business logic layer (also known as domain layer)
- Data access layer (also known as persistence layer)

Our application has the software divided into the following layers:

- Presentation Layer: KIVY UI Framework
- Application Layer: Python + HTML
- Logic Layer: Python + Encryption + Key Exchange (Diffie Hellman)

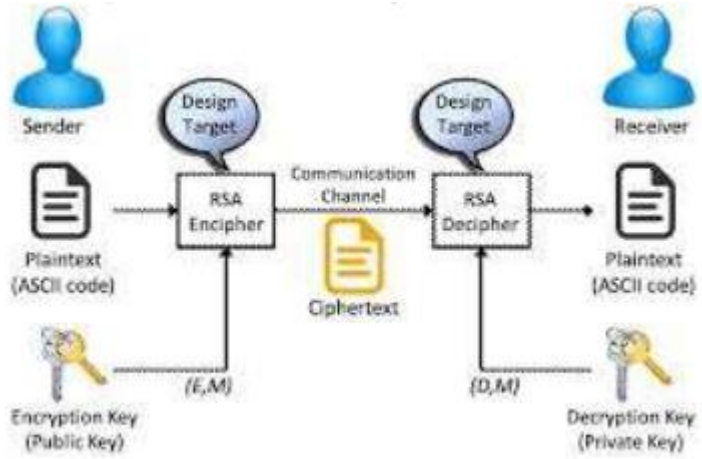
- Data Access Layer: Server sided Python Scripting



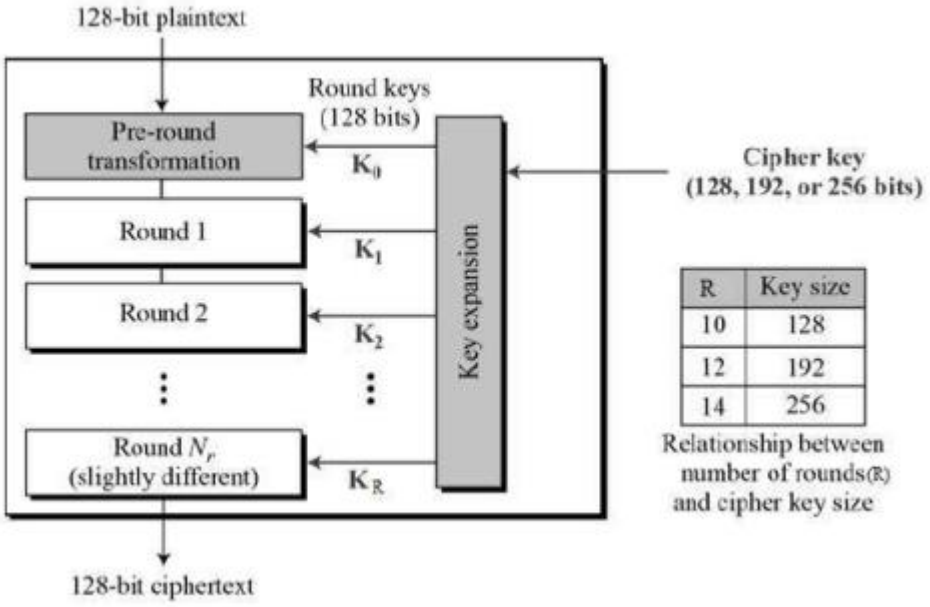
Algorithms and Flowcharts

The algorithms used in this project are as follows:

1. **RSA algorithm:** RSA is the algorithm used by modern computers to encrypt and decrypt messages. It is an asymmetric cryptographic algorithm. Asymmetric means that there are two different keys. This is also called public key cryptography, because one of them can be given to everyone. The other key must be kept private. It is based on the fact that finding the factors of an integer is hard (the factoring problem). RSA stands for Ron Rivest, Adi Shamir and Leonard Adleman, who first publicly described it in 1978. A user of RSA creates and then publishes the product of two large prime numbers, along with an auxiliary value, as their public key. The prime factors must be kept secret. Anyone can use the public key to encrypt a message, but with currently published methods, if the public key is large enough, only someone with knowledge of the prime factors can feasibly decode the message



2. AES algorithm: The more popular and widely adopted symmetric encryption algorithm likely to be encountered nowadays is the Advanced Encryption Standard (AES). AES is an iterative rather than Feistel cipher. It is based on 'substitution–permutation network'. It comprises of a series of linked operations, some of which involve replacing inputs by specific outputs (substitutions) and others involve shuffling bits around (permutations). Interestingly, AES performs all its computations on bytes rather than bits. Hence, AES treats the 128 bits of a plaintext block as 16 bytes. These 16 bytes are arranged in four columns and four rows processing as a matrix .



R	Key size
10	128
12	192
14	256

Relationship between number of rounds(R) and cipher key size

Common hash functions used

There are several hash functions that are widely used. All were designed by mathematicians and computer scientists. Over the course of further research, some have been shown to have weaknesses, though all are considered good enough for noncryptographic applications.

- **MD5**
  The MD5 hash function produces a 128-bit hash value. It was designed for use in cryptography, but vulnerabilities were discovered over the course of time, so it is no longer recommended for that purpose. However, it is still used for database partitioning and computing checksums to validate files transfers.

- **SHA-1**
  SHA stands for Secure Hash Algorithm. The first version of the algorithm was SHA-1, and was later followed by SHA-2 (see below).

  Whereas MD5 produces a 128-bit hash, SHA1 generates 160-bit hash (20 bytes). In hexadecimal format, it is an integer 40 digits long. Like MD5, it was designed for cryptology applications, but was soon found to have vulnerabilities also. As of today, it is no longer considered to be any less resistant to attack than MD5.

- **SHA-2**
  The second version of SHA, called SHA-2, has many variants. Probably the one most commonly used is SHA-256, which the National Institute of Standards and Technology (NIST) recommends using instead of MD5 or SHA-1.

  The SHA-256 algorithm returns hash value of 256-bits, or 64 hexadecimal digits. While not quite perfect, current research indicates it is considerably more secure than either MD5 or SHA-1.

  Performance-wise, a SHA-256 hash is about 20-30% slower to calculate than either MD5 or SHA-1 hashes.

- **SHA-3**
  This hash method was developed in late 2015, and has not seen widespread use yet. Its algorithm is unrelated to the one used by its predecessor, SHA-2.

  The SHA3-256 algorithm is a variant with equivalent applicability to that of the earlier SHA-256, with the former taking slightly longer to calculate than the later.

Code

Server Code: This is the basic code of the server of 185 lines.

```
ALLOWED_EXTENSIONS = set(['txt', 'pdf', 'png', 'jpg', 'jpeg', 'gif','aes'])
app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
def allowed_file(filename):
return filename[-3:].lower() in ALLOWED_EXTENSIONS
@app.route('/uploader', methods=['GET', 'POST'])
def upload_file():
if request.method == 'POST':
file = request.files['file']
idf = request.form['idx']
print(idf)
#div=idf.strip().split(',')
with open("xexe.txt", "a") as myfile:
```

```
myfile.write(idf+'\n')
if file and allowed_file(file.filename):
print ('**found file', file.filename)
filename = secure_filename(file.filename)
file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
# for browser, add 'redirect' function on top of 'url_for'
return url_for('upload_file')
return '''
@app.route('/')
def index():
if 'username' in session:
return 'Logged in as %s' % escape(session['username'])
return 'You are not logged in'
@app.route('/yolo')
def yolo():
if 'username' in session:
fobj = open("xexe.txt", "r")
#x=fobj.read()
k=[]
x=''
for i in fobj:
k=i
k=k.split(',')
print(k)
if (k[1]==(session['username']+'\n')):
x+=k[0]+'\n'
```

```
print(x)
fobj.close()
return 'Logged in as'+escape(session['username'])+x
return 'You are not logged in'
@app.route('/login', methods=['GET', 'POST'])
```

```
def login():
if request.method == 'POST':
session['username'] = request.form['username']
return redirect(url_for('selectfile'))
return '''
<form action="" method="post">
<p><input type=text name=username>
<p><input type=submit value=Upload>
</form>
'''
@app.route('/selectfile', methods=['GET', 'POST'])
def selectfile():
Tokenname='koala'
try:
Tokenname=session['username']
except: pass
fobj = open("xexe.txt", "r")
k=[]
x='~'
for i in fobj:
if (Tokenname=='koala'): break
```

```
k=i.strip().split(',')
if (k[2]==(Tokenname)):
x+= k[0] + ' ' + k[1] + '|'
x+='~'
if request.method == 'POST':
session['username'] = request.form['userauth']
filename= request.form['filename']
return redirect(url_for('download',filename=filename))
return x+'''<form action="" method="post">
<p><input type=text name=filename>
<p><input type=text name=userauth>
<p><input type=submit value=Login></form>'''
@app.route('/uploads/<filename>', methods=['GET', 'POST'])
def download(filename):
uploads = os.path.join(current_app.root_path,'uploads')
filename=filename[:filename.index('.')]+'.aes'
fobj = open("xexe.txt", "r")
k=[]
x=''
for i in fobj:
k=i.strip().split(',')
if (k[0]==filename):
break
print(k[1])
if(session['username']==k[2]):
return send_from_directory(directory=uploads, filename=filename)
```

```
else:
return send_from_directory(directory=uploads, filename='IMG.jpg')
```

```
@app.route('/logout')
def logout():
# remove the username from the session if it's there
session.pop('username', None)
return redirect(url_for('yolo'))
@app.route('/keys',methods=['GET','POST'])
def keys():
g = 7
n = 2860486313
z='~'+str(g)+','+str(n)+'~'
if request.method == 'POST':
keydata = request.form['keyfile'].split(',')
keyx={}
keyx[keydata[0]] = keydata[1]
file_Name = "testfile"
print(keyx)
fileObject = open(file_Name,'rb')
keyauth = pickle.load(fileObject)
keyauth.update(keyx)
fileObject.close()
fileObject = open(file_Name,'wb')
pickle.dump(keyauth,fileObject)
fileObject.close()
print(keyauth)
```

```
return url_for('keys')
return z+'''<form action="/keys" method="post">
<p><input type=text name=keyfile>
<p><input type=submit value=Login></form>'''
@app.route('/diffie',methods=['GET','POST'])
def diffie():
if request.method == 'POST':
getreq = request.form['key']
file_Name = "testfile"
fileObject
```

This is the client code of 281 lines

```
from kivy.uix.floatlayout import FloatLayout
from kivy.properties import ObjectProperty
from kivy.properties import StringProperty
from kivy.uix.popup import Popup
from kivy.factory import Factory
from kivy.graphics import Color,Rectangle
from kivy.core.window import Window
import requests
import pyAesCrypt
from os import stat
```

```
import os
import hashlib
import binascii
import urllib
import time
#AES256-CBC
class pr2kvlogin(FloatLayout):
username_text_input = ObjectProperty()
def login(self):
userx = self.username_text_input.text
Token = userx
url = url_base + "keys"
bToken=bytes(Token, 'utf-8')
dk = hashlib.pbkdf2_hmac('sha256', bToken, b'guessthis', 100000)
```

```
global trip
trip = (hashlib.md5(bToken).hexdigest())[-10:]
print(trip)
key=int(binascii.hexlify(dk),16)
a=int(str(key)[:6])
b=int(str(key)[-5:])
fp = urllib.request.urlopen(url)
mybytes = fp.read().decode("utf8")
fp.close()
x='';p=0
for i in mybytes:
if(i=='~'):p+=1;continue;
if(p==1): x+=i
if(p==2): break
x=list(map(int,x.strip().split(',')))
print(2)
G=x[0];N=x[1]
k=(((G**a)%N)**b)%N;
global Keys
Keys = [a,b,k,N]
print(3)
print(G)
LOCUS=trip+','+str(k)
data={'keyfile':LOCUS}
```

```
print(LOCUS)
r = requests.post(url,data=data)
print(4)
f = open('prmenu.kv','w')
xrite = "# Reference ClientA.py\n#: import main ClientA\nprmenu:\n<Button>:\n\tfont_size: 30\n\tcolor:1,1,1,1\n<FloatLayout>:\n\tAsyncImage:\n\t\tsource: 'back.jpg'\n\t\tcanvas:"
+ "\n\t\t\tColor:\n\t\t\t\trgba: 0.478, 0.478, 0.478,0.6\n\t\t\tRectangle:\n\t\t\t\tpos: 430,100\n\t\t\t\tsize: 350,400\n\t\t\tRectangle:\n\t\t\t\tpos:
```

```
50,100\n\t\t\t\tsize:
300,400\n\t\t\tRectangle:\n\t\t\t\tpos: 385,100\n\t\t\t\tsize: 10,400"
xrite+="\n\tLabel:\n\t\ttext: '"+trip+"'\n\t\tfont_size: 20\n\t\tpos: -
200,0\n\t\tsize: 10,400\n\tButton:\n\t\ttext:'Send'\n\t\tsize_hint: 0.3,
0.08\n\t\tpos:
490,250\n\t\ton_release: root.Sendscreen()\n\tButton:\n\t\ttext:
'Receive'\n\t\tsize_hint: 0.3, 0.08\n\t\tpos: 490,350\n\t\ton_press:
root.Receivescreen()"
f.write(xrite)
f.close()
time.sleep(2)
App.get_running_app().destroy_settings()
App.get_running_app().stop()
prmenuApp().run()
class pr2kvloginApp(App):
```

```
pass
class prmenu(FloatLayout):
def Sendscreen(self):
App.get_running_app().stop()
App.get_running_app().destroy_settings()
#self.root_window.close()
prSendApp().run()
def Receivescreen(self):
App.get_running_app().stop()
App.get_running_app().destroy_settings()
#self.root_window.close()
prReceiveApp().run()
class prmenuApp(App):
pass
class prReceive(FloatLayout):
filesel_text_input = ObjectProperty()
error_output = StringProperty()
file_output = StringProperty()
def __init__(self, **kwargs):
super(prReceive, self).__init__(**kwargs)
url = url_base + 'login'
data={'username':trip}
r = requests.post(url, data=data)
x='';p=0
global idlist
idlist={}
```

```
for i in r.text:
if(i=='~'):p+=1;continue;
if(p==1): x+=i
if(p==2): break
x=x.strip().split('|')
MArr=[]
```

```
for i in x:
if(len(i)!=0):
MArr.append(i)
print(MArr)
for i in MArr:
psp =i.split(' ')
idlist.update({psp[0]:psp[1]})
self.file_output = ''
for i in idlist.keys():
print(i)
self.file_output += str(i)+' - '
def receive(self):
filesel = self.filesel_text_input.text
if(filesel not in idlist.keys()):
self.error_output = 'Wrong File Selected'
else:
fname = filesel
url = url_base + "diffie"
sendto = idlist[fname]
data={'key':sendto}
```

```
r = requests.post(url,data=data)
idpub = int(r.text)
password = (((idpub**Keys[0])%Keys[3])**Keys[1])%Keys[3]
password = str(password)
print(password)
bufferSize = 64 * 1024
path = fname[:fname.index('.')]
format = '.'+fname[fname.index('.')+1:]
url = url_base + "selectfile"
data={'filename':fname, 'userauth': trip}
r = requests.post(url, data=data)
time.sleep(1)
open('filec/'+path+'.aes', 'wb').write(r.content)
print(path+format)
encFileSize = stat('filec/'+path+'.aes').st_size
# decrypt
with open('filec/'+path+'.aes', "rb") as fIn:
with open('filed/'+path+format, "wb") as fOut:
try:
pyAesCrypt.decryptStream(fIn, fOut, password, bufferSize, encFileSize)
except ValueError:
remove('filed/'+path+format)
class prReceiveApp(App):
pass
```

```
class prSend(FloatLayout):
sendto_text_input = ObjectProperty()
filepath_text_input = ObjectProperty()
error_output = StringProperty()
```

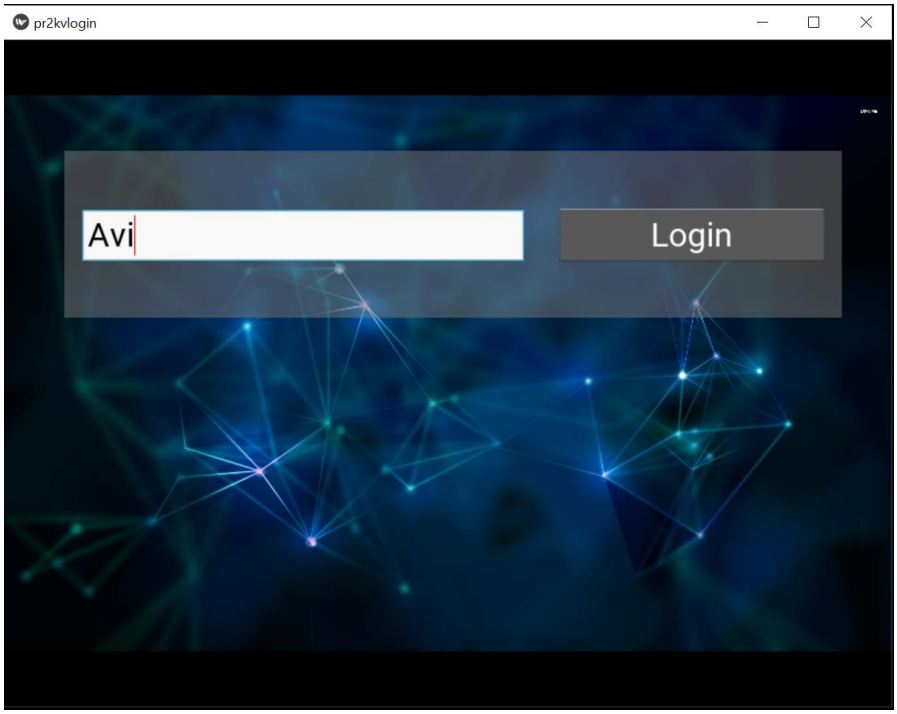
```
def __init__(self, **kwargs):
super(prSend, self).__init__(**kwargs)
def send(self):
sendto = self.sendto_text_input.text
FilePath = self.filepath_text_input.text
print(FilePath)
if(len(sendto)==0):
self.error_output = 'Enter Sender'
elif(len(FilePath)==0):
self.error_output = 'Select File'
else:
self.error_output = ''
url = url_base + "diffie"
data={'key':sendto}
r = requests.post(url,data=data)
idpub = int(r.text)
global Keys
#try:
password = (((idpub**Keys[0])%Keys[3])**Keys[1])%Keys[3]
#except:
# password = 12345
print(5)
```

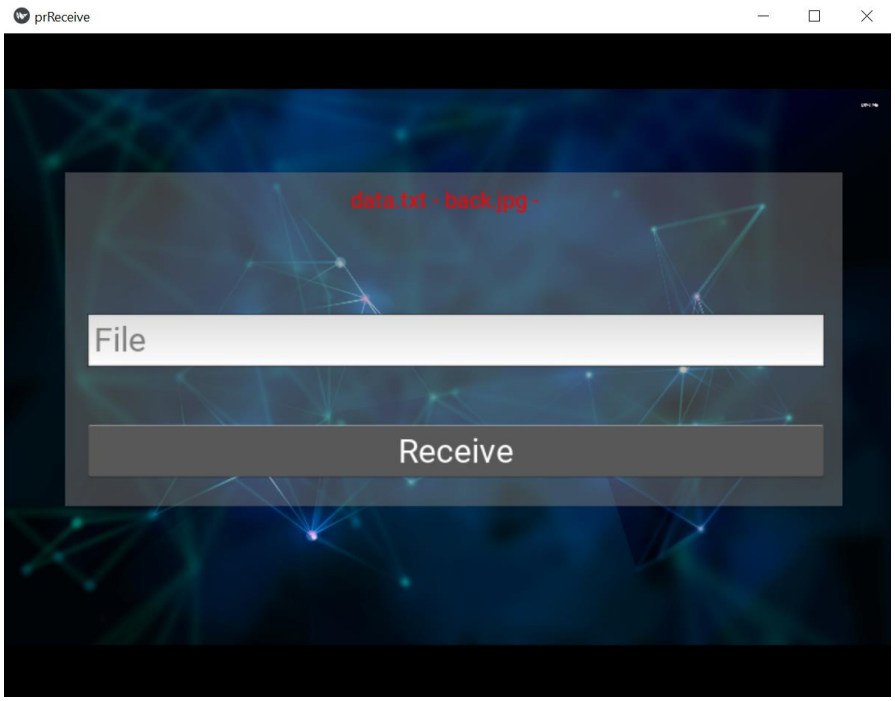
```
password=str(password)
bufferSize = 64 * 1024
path = FilePath
pathx=path[path.index('.')+1:]
with open("filea/"+path, "rb") as fIn:
with open("fileb/"+path[:path.index('.')]+".aes", "wb") as fOut:
pyAesCrypt.encryptStream(fIn, fOut, password, bufferSize)
fin=open("filea/"+path, "rb")
url = url_base + "uploader"
a=path + ',' + trip + ',' + sendto
fin = open('fileb/'+path[:path.index('.')]+'.aes', 'rb')
files={'file': fin}
data = {'idx':a}
print(a)
try:
r = requests.post(url,files=files, data=data)
finally:
fin.close()
print(password)
self.error_output = 'Success'
#time.sleep(10)
App.get_running_app().stop()
pr2kvloginApp().run()
class prSendApp(App):
```

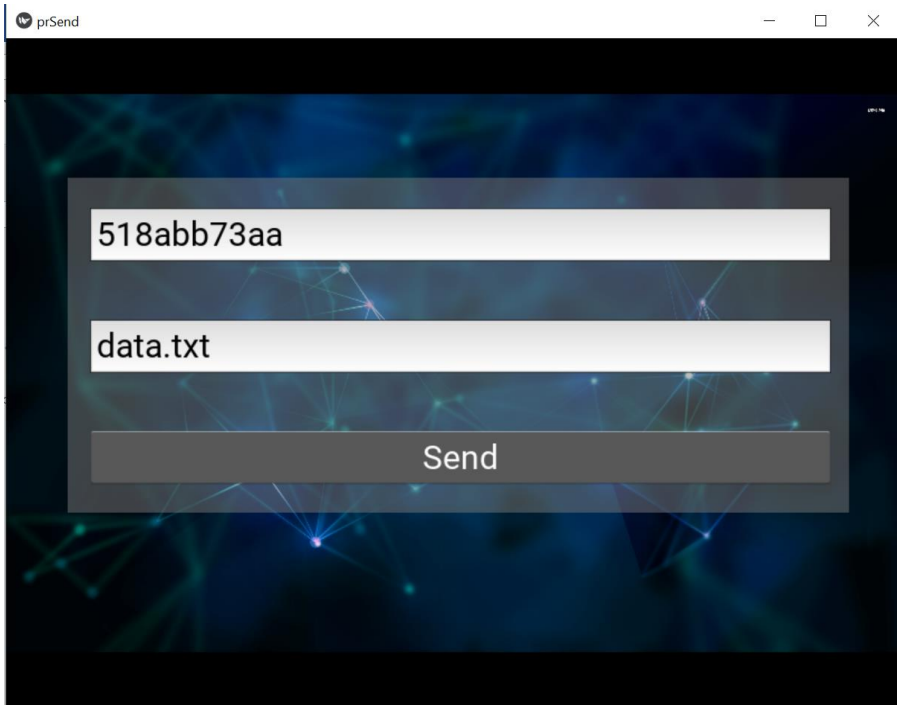
```
pass
if __name__ == '__main__':
url_base = 'http://127.0.0.1:5000/'
```

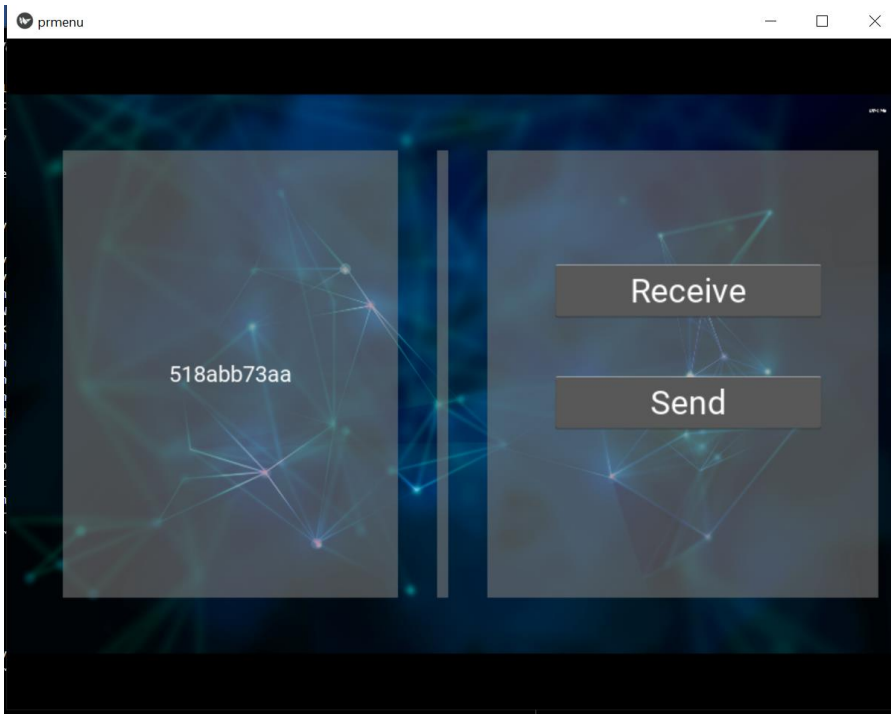
```
#global                                                    userx
pr2kvloginApp().run()
#prSendApp().run()
*
```

Results









THE END

THANK YOU MA'AM